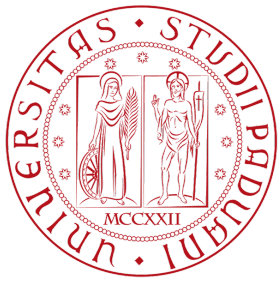

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Data-Driven Information Retrieval:
Riproducibilità di uno stemmer probabilistico

Relatore:
Prof.ssa Maristella AGOSTI

Laureando:
Giulio BUSATO

Correlatore:
Dr. Ing. Gianmaria SILVELLO

Anno Accademico 2016/2017

Indice

Elenco delle tabelle

1 Introduzione

La riproducibilità degli esperimenti è uno dei cardini della scienza moderna. In molte aree di ricerca, come ad esempio il reperimento dell'informazione (*Information Retrieval, IR*), la fase sperimentale ha sempre avuto un ruolo di fondamentale importanza. Le prestazioni di un sistema IR, infatti, dipendono strettamente dalla sua efficacia nel recuperare l'informazione ritenuta rilevante, ma a causa della natura non deterministica di questo aspetto, l'unico vero mezzo per valutarne le prestazioni è la valutazione sperimentale. Essendo l'IR una disciplina tradizionalmente radicata nella sperimentazione, ha per questa ragione avuto sempre tra gli obiettivi primari la riproducibilità degli esperimenti.

1.1 Obiettivi della tesi

L'obiettivo di questa tesi è studiare la riproducibilità di un algoritmo di stemming basato su tecniche probabilistiche. In particolare, vengono riprodotti sia il codice relativo all'algoritmo che gli esperimenti effettuati in fase di training e test, i cui risultati originali sono riportati nell'articolo nel quale questo approccio è stato presentato.

L'algoritmo in esame, denominato *GRAS (GRAph-based Stermmer)*, è un soluzione efficiente per effettuare lo stemming di vocaboli, pensato in

particolar modo per i sistemi di Information Retrieval. L'approccio di questa soluzione si basa sull'analisi della frequenza delle coppie di suffissi tra due parole che condividono un prefisso comune. Sfruttando tali informazioni, viene costruito un grafo che collega parole morfologicamente correlate, attraverso degli archi il cui peso è la frequenza delle coppie di suffisso associate ad esse. Attraverso una misura definita coesione, è possibile stabilire quanto le parole risultino morfologicamente correlate e quindi se appartengono alla stessa classe lessicale, a cui viene associato il relativo stem.

Il lavoro consiste innanzitutto nel riprodurre l'implementazione dell'algoritmo, basandosi sui concetti espressi in fase di presentazione dell'approccio e sullo pseudocodice a disposizione.

La seconda parte è relativa alla riproduzione degli esperimenti di training e test, con l'obiettivo di ottenere gli stessi risultati riportati dagli autori di GRAS, giustificando la natura delle eventuali incongruenze riscontrate. Tali esperimenti sono stati realizzati con lo scopo di misurare l'efficacia dell'approccio proposto, attraverso l'uso di metodi di valutazioni indiretti, cioè valutando il miglioramento delle prestazioni del reperimento, indotto dall'uso dello stemmer in fase di indicizzazione e interrogazione.

In questa fase sarà fondamentale evidenziare dove gli autori sono stati imprecisi nel riportare i passi svolti, o approssimativi nella definizione dei parametri utilizzati per arrivare ai risultati che hanno riportato. Uno degli aspetti che risulta fondamentale nel contesto della riproducibilità di un esperimento, è proprio il tuning dei parametri.

1.2 Struttura e organizzazione dei capitoli

La tesi è strutturata come segue. Il capitolo 2 riguarda le motivazioni che hanno spinto alla realizzazione di questo lavoro. Nella prima parte viene introdotto il concetto di Data-Driven Information Retrieval, ovvero il nuovo

approccio a questa disciplina, sempre più focalizzato sui dati; nella seconda parte, invece, vengono presentati gli aspetti salienti della riproducibilità, con particolare attenzione a che cosa significa riproducibilità nel contesto dell'IR.

Nel capitolo 3, invece, viene introdotto il concetto di stemming, dandone la definizione, evidenziando il ruolo che ha all'interno del processo di reperimento e introducendo alcuni concetti base di linguistica per illustrare il funzionamento di uno stemmer. Nella seconda parte del capitolo vengono descritte le varie tipologie di stemmer e vengono elencate le differenze in particolare per gli stemmer basati su regole e per quelli che utilizzano metodi probabilistici, di cui GRAS è un esempio. Vengono infine descritti vari metodi di valutazione diretta e indiretta di uno stemmer, questi ultimi vengono poi utilizzati nella fase sperimentale i cui risultati sono riportati nel capitolo 5.

Il capitolo 4 è invece interamente dedicato all'algoritmo GRAS. Viene prima di tutto presentato l'approccio di questa soluzione, evidenziandone i concetti su cui si basa il metodo di riconoscimento delle classi di parole morfologicamente correlate. Nella seconda parte vengono descritti in dettaglio i punti chiave dell'algoritmo, in particolare l'identificazione delle coppie di suffisso frequenti, la costruzione del grafo e il riconoscimento delle classi morfologiche, riportando lo pseudocodice relativo. Nell'ultima parte del capitolo vengono fatte alcune considerazioni sui parametri da cui dipendono le prestazioni dello stemming realizzato dall'algoritmo.

Nel capitolo 5 viene presentato lo studio della riproducibilità di GRAS. Nella prima parte viene descritta la fase riguardante l'implementazione dell'algoritmo proposto, riportando le parti più interessanti con il relativo codice, giustificando le scelte progettuali e le strutture dati utilizzate. Dopo aver brevemente descritto gli strumenti e le collezioni sperimentali utilizzati, si passa alla seconda parte, relativa alla riproducibilità della fase sperimentale, ovvero agli esperimenti effettuati per il training e il test dello stemmer. In questa parte vengono forniti in maniera dettagliata i passaggi svolti e le

scelte effettuate per produrre i risultati ottenuti, confrontandoli con quelli di riferimento relativi all'articolo di GRAS.

Il capitolo 6, infine, presenta le conclusioni che si possono trarre a seguito del lavoro svolto.

2 Motivazioni e stato dell'arte

2.1 Data-Driven Information Retrieval

Fin dal principio, il repermento dell'informazione è stata una disciplina focalizzata su un scopo chiaro e ben preciso: fornire documenti rilevanti, al fine di soddisfare un'esigenza informativa di un utente. Questo è sempre stato anche l'obiettivo alla base dei sistemi di reperimento, dei modelli e degli algoritmi fondamentali di questa materia.

Nel corso dei decenni, quest'area di ricerca è stata in grado di reagire ad almeno tre principali cambiamenti:

1. la standardizzazione dei dati con cui effettuare i test, attraverso la creazione di collezioni specifiche, molto simili nei contenuti e nelle dimensioni a quelle reali, per poter testare in modo uniforme l'efficacia dei modelli e dei sistemi sviluppati;

2. la variazione da collezioni di dati omogenei a collezioni eterogenee, esplorando così nuove aree e aprendo a nuovi ambiti di ricerca, come'è avvenuto ad esempio con il Multimedia Information Retrieval;

3. la necessità di avere metodi in grado di gestire ed elaborare collezioni di dati derivanti da esperimenti scientifici, che generano notevoli flussi di dati in brevi intervalli di tempo.

2.1.1 L'evoluzione dei sistemi di reperimento

Con il passare degli anni e con il progresso scientifico di questa disciplina, anche i sistemi di reperimento si solo evoluti. Oggi, possono essere considerati come veri e propri agenti intelligenti che mirano a svolgere la migliore azione possibile, con lo scopo di fornire le informazioni giuste alle persone giuste nel modo giusto [Agosti et al., 2016].

Queste azioni, possono spaziare dalla comprensione e l'interpretazione della richiesta informativa dell'utente, che spesso è molto più articolata e complessa di quello che viene espresso nelle query, fino alla ricerca delle fonti d'informazione più appropriate, per poter fornire risultati coerenti rispetto al contesto di ricerca, presentandoli secondo un certo grado di rilevanza. Inoltre, ognuna di queste azioni può influenzare sia il comportamento dell'utente, che quello futuro del sistema. I sistemi di reperimento dell'informazione infatti, sono ormai in grado di apprendere da quanto fatto in precedenza, migliorando volta per volta nell'efficacia e interpretando i segnali comportamentali degli utenti in relazione alle risposte fornite.

Alcuni esempi concreti possono essere il suggerimento di una query da parte del sistema, prevedendo quindi quello che l'utente potrebbe voler cercare; metodi basati sull'interleaving possono invece far dedurre in modo affidabile le preferenze di un utente; l'analisi del click-through rate, invece, permette di aumentare la rilevanza di certi risultati per le query molto frequenti (head queries); attraverso l'analisi dei log o dello storico delle sessioni, è possibile capire altri aspetti e comportamenti dell'utente.

Questo cambiamento, che ha portato i sistemi di IR a prendere sempre più le vesta di agenti intelligenti sopra descritti, è stato incentivato negli ultimi anni principalmente da due fattori:

- la crescente disponibilità di dati relativi all'interazione tra utenti e sistemi: dati che sono in grado di catturare, seppure ad un certo livello

di astrazione, informazioni relative alle richieste degli utenti, alle loro preferenze o al feedback in funzione dei risultati presentati;

- i progressi nella capacità di analisi e interpretazione di questi dati, grazie alle sempre più diffuse ed efficaci tecniche di machine learning, data mining e knowledge discovery che utilizzano metodi di apprendimento a supervisione limitata o addirittura senza supervisione.

2.1.2 Approccio Data-Driven

L'elevata disponibilità di ingenti dataset e lo sviluppo delle tecniche di machine learning, in combinazione ai progressi tecnologici per quanto riguarda il calcolo ad alte prestazioni e capacità di storage, di cui spesso si ignora l'importanza cruciale, hanno portato ad un cambiamento radicale nell'IR nel corso degli ultimi anni, anche da un punto di vista metodologico.

L'Information Retrieval, è stata tradizionalmente una disciplina problem-driven, in cui chi si occupava di fare ricerca, progettava e sviluppava metodi e modelli adatti a risolvere un problema derivante da una specifica esigenza. Progressivamente si è spostato il baricentro verso un approccio data-driven, in cui il punto di partenza non è più necessariamente il modello matematico con il quale vengono gestiti i dati, ma viceversa, partendo da un insieme di dati si modella il problema o addirittura si ottengono spunti da utilizzare per nuove soluzioni. Per i ricercatori di questo settore, quindi, diventa di vitale importanza possedere abilità e competenze trasversali per quanto riguarda la gestione e l'elaborazione di grandi moli di dati, oltre alla capacità analitica di interpretarli, al fine di ideare e sperimentare nuove tecniche [Abadi et al., 2016].

I vantaggi che ha portato questo nuovo tipo di approccio sono davvero molti. Alcune soluzioni allo stato dell'arte di task tradizionali dell'IR come ad esempio ranking, user modeling, recommendation, document e

query understanding, si basano su tecniche di apprendimento automatico e raggiungono il massimo in termini di prestazioni quando hanno a disposizione grandi quantità di dati su cui eseguire l'addestramento. Inoltre, come diretta conseguenza, si hanno a disposizione nuove soluzioni scalabili anche per task che in precedenza risultavano troppo complessi per essere trattati dai tradizionali sistemi di IR, come ad esempio analisi e comprensione di immagini o la misura della similarità semantica di testi.

Non è tuttavia corretto pensare che la strada sia soltanto in discesa, poiché i dati raccolti e le informazioni acquisite dalla loro analisi, in per sé, non garantiscono che ci sia un valore aggiunto; tutto sta nell'interpretarli e nell'utilizzarli nella maniera più corretta possibile, per fare in modo che ci sia un effettivo e corretto utilizzo di quanto dedotto.

Un altro aspetto particolarmente delicato e da non sottovalutare, che è strettamente correlato alla raccolta e l'utilizzo dei dati derivanti dell'interazione tra uomo e macchina, è quello relativo alla privacy. L'utilizzo eccessivo o troppo invasivo di informazioni relative a preferenze e comportamenti dell'utente possono interferire con quello che è lo scopo primario dell'approccio, ovvero migliorare la qualità del servizio, suscitando negli utilizzatori un effetto controproducente.

È inoltre indispensabile considerare che, sia la raccolta che l'analisi dei dati sono soggette ad errori ed alterazioni del contenuto informativo a causa dei dati inesatti, mancanti o condizionati da fattori esterni, che introducono rumore e distorsione in quella che è la realtà che vanno a modellare.

Questo nuovo approccio data-driven, lascia davanti a sé grandi prospettive e interessanti sfide nel campo della ricerca nell'Information Retrieval, alcune delle quali sono:

- la definizione di un data-stack di riferimento e ricerca di nuove soluzioni algoritmiche. La dimensione e la complessità dei dati impone che siano adottate soluzioni sempre più standardizzate e robuste, in modo

da semplificare e rendere più scalabili le fasi di raccolta, scrematura, convalida e utilizzo dei dati su larga scala;

- incentivare, rendendo sempre più efficiente e conveniente l'uso dei dati già disponibili. Il riutilizzo di dataset infatti, permette di abbassare i costi ed avere risultati confrontabili per le diverse soluzioni testate;
- essere consapevoli della provenienza e della modalità con cui i dati a disposizione sono stati generati può essere di fondamentale importanza per evitare o quantomeno mitigare gli errori, le distorsioni cognitive e le questioni etiche che potrebbero influenzare l'interpretazione dei dati e dei modelli generati;
- rendere i modelli sempre più trasparenti ed espliciti per quanto riguarda il loro comportamento interno, in contrapposizione all'approccio black-box che fino ad ora ha prevalso.

2.1.3 Data Stack nell'Information Retrieval

A questo proposito è stato proposta una possibile formulazione di Data Stack per l'Information Retrieval [Alonso, 2016], nel quale in ogni livello, che è associato ad una fase del processo di reperimento, vengono riportati i rispettivi task e le tipologie di dati che si presentano o vengono elaborati:

1. **Acquisizione:** la prima fase prevede l'acquisizione e l'elaborazione dei dati grezzi, che comprende la pulizia, la scrematura e l'eliminazione di dati duplicati o non conformi.
2. **Aumento dell'espressività**: l'obiettivo di questa fase è aumentare l'espressività semantica dei dati acquisiti attraverso l'aggiunta di annotazioni, informazioni addizionali e metadati. I task relativi a questa fase sono: named-entity detection, information extraction, metadata generation, content classification.

Fase	Dati	Task
Acquisizione	dati grezzi	near-duplicate detection, crawling, data cleaning
Aumento dell'espressività	annotazioni, metadati, informazioni semantiche	named-identity detection, information extraction, metadata generation, content classification
Indicizzazione	indici e giudizi di rilevanza	indexing, relevance ranking, feature engineering
Analisi comportamentale	dati comportamentali relativi all'utente	search query logs, link sharing, click rate analisys
Analisi sperimentale	dati generati dalle sperimentazioni	crowdsourcing, interleaving, sampling, A-B testing

Tabella 2.1. *Information Retrieval Data Stack*

3. **Indicizzazione:** in questa fase viene effettuata l'indicizzazione attraverso efficienti strutture dati.

4. **Analisi comportamentale** l'obiettivo di questa fase è quello di catturare i dati relativi all'attività e al comportamento dell'utente, ovvero tutte quelle informazioni che possono essere utili ad aumentare le performance del sistema, come ad esempio lo studio dei query logs, click, link sharing e dei dati relativi alla sessione.

5. **Analisi Sperimentale** questa ultima fase, riguarda l'analisi, la valutazione, l'esplorazione e l'utilizzo dei dati raccolti nei livelli precedenti. Le tecniche utilizzate in questa fase possono essere A-B testing, interleaving, sampling, crowdsourcing ed altre ancora.

L'idea che sta alla base di questo data-stack, è presentare e far luce su quelle che sono le opportunità che potrebbero presentarsi concentrandosi sui diversi

livelli, e quindi sul tipo di informazioni che si può ottenere esplorandone ed analizzandone i dati.

2.2 Riproducibilità

La riproducibilità è sempre stato un aspetto primario nel campo della ricerca scientifica. La possibilità di riprodurre un esperimento, è infatti uno dei pilastri su cui si fonda il metodo scientifico sperimentale introdotto da Galileo ormai molti secoli fa e su cui si basa ancora oggi la scienza moderna.

Negli ultimi anni, essendosi molte discipline scientifiche sempre più focalizzate nell'aggregazione e l'analisi dei dati sperimentali, l'importanza di questo aspetto ha ottenuto ancora più rilevanza nelle varie comunità di ricerca.

2.2.1 L'importanza di avere risultati riproducibili

In molte aree scientifiche di ricerca, in particolar modo nel settore dell'informatica, la fase sperimentale ha un ruolo di fondamentale importanza. Oltre agli aspetti teorici degli algoritmi e dei metodi utilizzati, l'efficacia e le prestazioni di un modello possono essere misurate soltanto tramite la sperimentazione. Nella maggior parte dei casi, i risultati sperimentali sono fortemente influenzati dai dati che ricevono in input, dalla modellazione dei parametri forniti al sistema, e dalle caratteristiche dell'ambiente di calcolo in cui gli esperimenti vengono eseguiti.

Purtroppo, ancora oggi molti esperimenti e valutazioni sperimentali vengono descritti in maniera approssimativa e non dettagliata, riportando in modo informale e incompleto le viarie fasi svolte e le specifiche utilizzate. Addirittura in alcuni casi anche i risultati finali sono presentati con poco rigore, in grafici o tabelle non semplici da interpretare, talvolta sprovvisti di spiegazioni e informazioni aggiuntive. Per concludere, anche il codice che ha prodotto i risultati spesso è non disponibile o difficilmente accessibile.

Tutto questo porta a gravi implicazioni per il progresso scientifico, e nei casi più estremi induce a una sorta di isolamento dei risultati da quello che è il contesto di ricerca [Freire et al., 2016]. Un'inadeguata documentazione, la presentazione non chiara dei risultati e l'inaccessibilità diretta alle risorse utilizzate, sia in termini di codice che di dati utilizzati per arrivare ad un determinato risultato, rende l'esperimento non replicabile e quindi non generalizzabile. Questo comporta che i risultati ottenuti non possano essere riconsiderati ed eventualmente estesi da lavori futuri. Oltre ad ostacolare il riuso effettivo del lavoro prodotto, questo limita l'impatto scientifico del risultato stesso e ovviamente ne compromette il rilievo, mettendone anche in discussione l'affidabilità e la validità dei risultati.

2.2.2 Riproducibilità dell'IR

Le prestazioni dei sistemi di reperimento non son determinate solo dalla loro efficienza, ma anche e soprattutto dalla loro efficacia, cioè la capacità di recuperare le risorse informative ritenute interessanti, presentandole secondo il corretto grado di rilevanza per l'utente, e allo stesso tempo evitando il recupero di quelle ritenute non rilevanti.

A causa della presenza di numerosi fattori non strettamente deterministici, come ad esempio il concetto di rilevanza per l'utente, il bisogno informativo dello stesso, che spesso è vago e mal formulato, o ancora la presenza di fonti di informazione non strutturate, l'unico mezzo per valutare le prestazioni dei sistemi di reperimento dell'informazione dal punto di vista dell'efficacia, è la valutazione sperimentale. L'Information Retrieval è infatti una disciplina tradizionalmente radicata nella sperimentazione e per questo ha sempre avuto tra gli obiettivi primari la ripetibilità degli esperimenti [Ferro et al., 2016].

La valutazione sperimentale nell'IR si basa sul paradigma di Cranfield, sviluppato agli inizi della seconda metà del '900 da Cyril Cleverdon, un bibliotecario del College of Aeronautics a Cranfield, in Inghilterra. Questo

paradigma, prevede l'uso di collezioni sperimentali, formate da insiemi di documenti selezionati in modo da essere rappresentativi sia in termini di argomento che in termini di quantità, ed un insieme di giudizi di rilevanza che, per ogni interrogazione effettuata sulla collezione, indicano quali documenti sono ritenuti rilevanti e quali no, per quel determinato argomento.

Al fine di incentivare l'utilizzo di questo metodo e di unificare gli sforzi per la creazione di collezioni sperimentali standard sempre più rappresentative della realtà su cui effettuare la valutazione, sono sorte delle apposite campagne per la generazione di collezioni sperimentali [Ferro, 2017]. A partire da TREC[1] (Text REtrieval Conference) organizzata nel 1992 dal NIST[2] (National Institute of Standards and Technology) negli USA, c'è stato un susseguirsi di campagne internazionali come NTCIR[3] (NII Testbed and Community for Information access Research) per il Giappone e l'Asia nel 1999, CLEF[4] (Conference and Labs of the Evaluation Forum), per quanto riguarda le lingue europee, a partire dal 2000 e ancora FIRE[5] (Forum for Information Retrieval Evalutation) in India, dal 2008, ognuna delle quali ha lo scopo di definire collezioni specifiche per determinati ambiti o lingue, o task dell'IR.

Un altro aspetto di rilievo per quello che riguarda la riproducibilità nell'IR è quello relativo alle system runs, ovvero i risultati che produce l'esecuzione di un sistema di reperimento. Poterli mettere a confronto facilmente permette di capire molti aspetti di un sistema o un modello, specialmente se si tratta di qualcosa di innovativo. Spesso anche l'uso dello stesso dataset e degli stessi strumenti software, non garantiscono di ottenere gli stessi risultati poiché vi sono una serie di parametri da calibrare che influenzano in maniera determinante i risultati finali. Questo si complica ancor di più quando parte dei dati sono relativi all'interazione tra l'utente e il sistema.

[1] http://trec.nist.gov/
[2] http://www.nist.gov/
[3] http://research.nii.ac.jp/ntcir/index-en.html
[4] http://www.clef-campaign.org/
[5] http://fire.irsi.res.in/fire/2016/home

Sebbene l'IR abbia una rinomata tradizione per quanto riguarda il rigore scientifico nella produzione dei dati sperimentali, non si può purtroppo dire lo stesso per quanto riguarda la gestione e la cura dei dati di rilievo e questo comporta inevitabilmente un ostacolo per quanto riguarda la riproducibilità [Ferro and Silvello, 2017b].

Non esiste infatti un unico formato standard per la modellazione e la descrizione dei dati sperimetali, così come vi è una certa carenza di metadati, che sarebbero utili per aumentarne l'espressività attraveso l'uso di annotazioni. Inoltre, in molte casi anche la semantica dei dati non è perfettamente chiara ed esplicita ed è necessario elaborarli con script la cui documentazione è spesso assente.

2.2.3 Uno sguardo al futuro

Sebbene negli ultimi anni si sia rivolta maggiormente l'attenzione su questo argomento, la riproducibilità rimane ancora un ambito su cui lavorare e in cui la ricerca può fare molti progressi, soprattutto dal punto di vista pratico.

Un esempio riguarda l'introduzione di specifiche infrastrutture che garantiscano la condivisione dei dati, e che ne permettano il riuso in modo facile ed accessibile. A questo proposito sta prendendo forma il concetto di "data citation" [Silvello and Ferro, 2016], ovvero la possibilità di fare riferimento diretto a dataset utilizzati in un determinato esperimento, o la possibilità di collegare direttamente un articolo scientifico ai dati che si sono utilizzati per raggiungere quel risultato.

Altri importanti aspetti riguardano lo sviluppo di sistemi di IR open-source, il cui incentivo all'utilizzo possa garantire la massima accessibilità a chiunque, così come alla creazione di strumenti di valutazione come servizio (Evaluation-as-a-Service) o ancora la definizione di baseline [Lin et al., 2016], che risultano fondamentali per quanto riguarda la riproducibilità di un risultato.

3 Stemming

3.1 Definizione di stemming

Lo stemming è un processo linguistico che ha lo scopo di eliminare le variazioni morfologiche di una parola, portandola a quella che è la sua forma base. Queste variazioni possono essere principalmente di tre tipi, ovvero flessioni, derivazioni e concatenazioni.

Contrariamente a quanto si possa pensare, lo stemming non è un argomento che interessa soltanto a chi opera nel campo della linguistica, ma ha un ruolo rilevante anche in molte discipline attinenti all'informatica, tra cui Information Retrieval, Natural Language Processing e Language Modeling [Singh and Gupta, 2016]. Molte applicazioni di queste aree di ricerca utilizzano nella fase di preprocessing sofisticati algoritmi di stemming, basati su tecniche di vario genere. Lo sviluppo di nuove soluzioni sempre più efficienti coinvolge quindi sia i ricercatori nel campo della linguistica, sia quelli di alcune discipline scientifiche.

La sfida non è banale, perché se da un lato l'obiettivo è quello di avere algoritmi veloci, in grado di elaborare in un tempo ragionevole vocabolari molto estesi, dall'altro è fondamentale anche che il risultato prodotto sia conforme alle regole linguistiche della specifica lingua trattata.

Quest'ultimo aspetto infatti è di particolare importanza: ogni lingua ha

le proprie regole di variazione morfologica e necessita che vengano adottate delle soluzioni specifiche basate su tali regole. A questo si aggiunge il fatto che esistono svariati alfabeti e particolari simboli fonetici per varie lingue esistenti, ognuna delle quali ha le proprie caratteristiche e peculiarità.

Tutto ciò fa ben intendere quanto lavoro richieda la realizzazione di tecniche automatiche per eseguire lo stemming, e perché questo sia ad oggi oggetto di ricerca. Quello a cui si mira, è ottimizzare le tecniche già note e proporne altre sempre più innovative, basate su modelli probabilistici che rendono gli algoritmi indipendenti dalla lingua trattata e sfruttano maggiormente le capacità di elaborazione dei calcolatori.

In questo capitolo verrà inizialmente presentato il ruolo che ha lo stemmig nell'Information Retrieval, evidenziandone i vantaggi che introduce l'uso di questa tecnica. Successivamente verranno introdotti alcuni concetti di base per comprendere meglio cosa realmente fa uno stemmer e saranno quindi presentate le principali tipologie di algoritmi di stemming. Per concludere, verranno illustrati i metodi di valutazioni utilizzati per misurarne le prestazioni.

3.2 Il ruolo dello stemming nell'IR

In Information Retrieval, lo stemming viene eseguito nella fase di preprocessing dei documenti all'interno del processo di indicizzazione.

Nella fase preliminare di questo processo viene effettuata l'analisi lessicale dei documenti, nella quale vengono estratti i token, ovvero tutti gli insiemi di caratteri delimitati da un separatore. Successivamente vengono rimosse le stopword, cioè tutte quelle parole che sono molto frequenti ma il cui contenuto informativo non è rilevante. Solitamente sono articoli, congiunzioni, preposizioni, pronomi e vengono elencate nelle apposite stoplist, che ovviamente variano a seconda della lingua considerata. Dopo

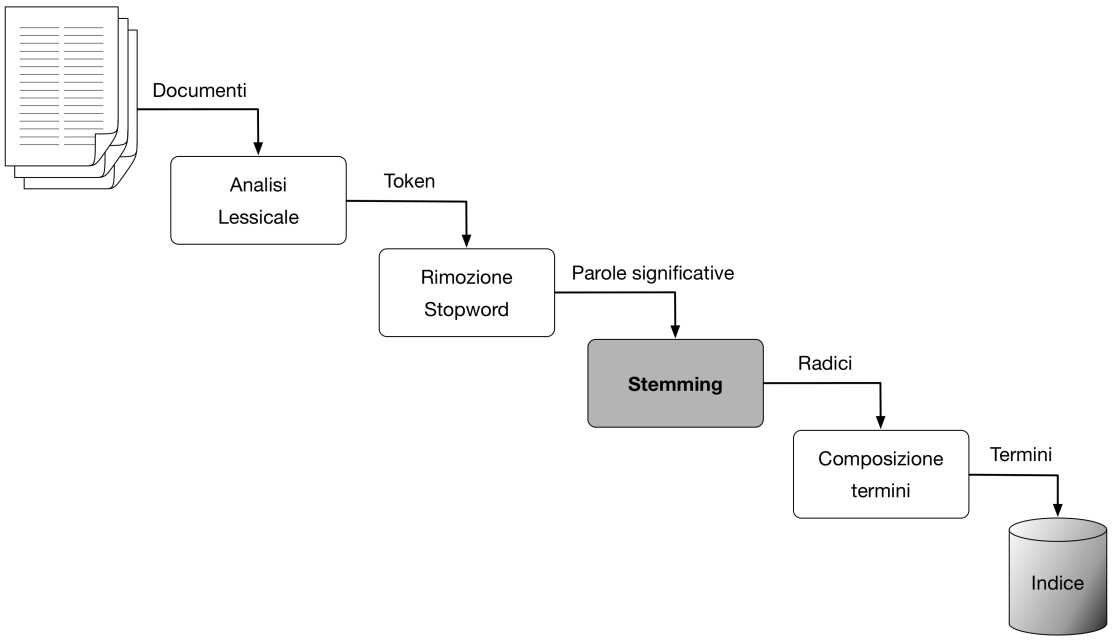

Figura 3.1. *Fasi del processo di indicizzazione*

aver rimosso le stopword si passa alla fase di stemming, in cui le parole vengono raggruppate nelle rispettive radici linguistiche, eliminando così le variazioni morfologiche. Il successivo step è relativo alla composizione dei termini e alla formazione di gruppi di parole. Alcuni termini infatti, se raggruppati, migliorano l'espressività del concetto associato o in alcuni casi esprimono un concetto differente dalle singole parole che lo compongono.

C'è da specificare il fatto che tutte queste fasi preliminari non sono obbligatorie, ad eccezione dell'analisi lessicale, che deve necessariamente essere eseguita per estrarre i token dai documenti. Le altre fasi sono facoltative e dipendono da quanto si vuole ottimizzare l'indice e dal tipo di sistema di reperimento che si vuole realizzare.

È inoltre fondamentale che lo stesso prepocessing eseguito sui documenti, venga eseguito di pari passo alle query con cui viene interrogato il sistema, altrimenti i vantaggi apportati da queste tecniche in fase di indicizzazione vengono del tutto vanificati.

Nei sistemi di IR, lo stemming viene utilizzato principalmente per due ragioni, entrambe delle quale portano ad un miglioramento delle prestazioni:

- L'uso dello stemming nella fase di indicizzazione permette di incrementare la precisione e ancor di più il richiamo [Kraaij and Pohlmann, 1996] in fase di reperimento, quindi in generale migliora le prestazioni del sistema da un punto di vista di efficacia. Eliminando le variazioni morfologiche di una parola, si diminuisce la possibilità che non ci sia corrispondenza tra i vocaboli presenti in un documento e quelli utilizzate dall'utente per formulare la query. Essi infatti potrebbero non essere perfettamente coincidenti, ma potrebbero riferirsi allo stesso concetto.

- L'uso dello stemming permette di diminuire sensibilmente la dimensione del dizionario dei termini, e quindi anche dell'indice che viene gestito dal sistema. Questo permette di avere strutture dati meno pesanti ed aumentare così le prestazioni in termini di efficienza. Effettuando lo stemming infatti, tutti i vocaboli relative alle differenti variazioni morfologiche di una parola, vengono associate all'unica radice da cui derivano, riducendo drasticamente il numero di termini da memorizzare. Ovviamente questo aspetto dipende fortemente dalla tipologia di stemmer che viene utilizzato, a seconda che usi una tecnica più o meno aggressiva.

Da un altro punto di vista, lo stemming può anche essere visto come un meccanismo di disambiguazione dei vocaboli, in quanto può normalizzare le variazioni di una parola fondendole in un unico termine che generalizza il concetto espresso [Krovetz, 1993]. Questo tuttavia è vero solo in parte, poiché ci sono molte parole che esprimono lo stesso concetto ma che non sono morfologicamente legate.

Le varie teniche di stemming risultano particolarmente efficaci con le lingue morfologicamente articolate, perché essendoci molte variazioni delle stessa parola, il vantaggio dall'accomunarle in un unico termine risulta più evidente. L'inglese, ad esempio, è una lingua poco strutturata dal punto di vista morfologico, ed i vantaggi che porta l'uso di questa tecnica risultano meno evidenti rispetto invece alle altre lingue europee. Ad esempio per per il francese, il portoghese e l'ungherese, che risultano essere molto strutturate, è stato dimostrato che l'effetto dello stemming porta a grossi vantaggi in termini di precisione e accuratezza dei sistemi di retrieval [Savoy, 2006].

3.3 Concetti preliminari

Di seguito verranno elencati alcuni concetti basilari della linguistica, in modo da poter comprendere meglio come uno stemmer opera e fornire il lessico di base per avere chiarezza sui metodi di stemming che verranno illustrati in seguito:

- *stem*: o anche radice, è la componente linguistica irriducibile che esprime il significato principale della parola. E' quindi la forma base di un vocabolo e può essere modificata con l'aggiunta di affissi (prefissi o suffissi) o la combinazione di più termini;

- *lessema* e *lemma*: i lessemi sono tutte le possibili forme e variazioni che una parola può assumere, ed hanno tutti lo stesso significato. Un lemma invece è una specifica forma che è stata definita per rappresentare tutti i lessemi, è la parola che si può trovare per esempio in un dizionario per esprimere il significato di quel concetto. Per i verbi italiani, il lemma corrisponde alla forma all'infinito, per i sostantivi invece il singolare maschile o femminile, ma questa convenzione varia a seconda della lingua;

- *flessione morfologica*: è un tipo di variazione linguistica apportata per caratterizzare i tratti grammaticali e sintattici di un vocabolo, senza alterarne il significato. Questa variazione è realizzata effettuando la coniugazione o la declinazioni della parola, che generalmente comporta l'aggiunta di affissi, per fare in modo che la frase risulti grammaticalmente corretta;

- *derivazione morfologica*: questo tipo di variazione linguistica, invece, comporta la modifica del significato della parola di partenza, generando quindi vocaboli differenti. Generalmente anche questo tipo di modifica viene effettuata tramite l'aggiunta di affissi, ma ha un risultato completamente differente rispetto alla precedente per quanto riguarda il significato della parola modificata;

- *stemming leggero* e *aggressivo*: questi due termini sono utilizzati per misurare il grado efficacia di uno stemmer. In particolare gli stemming leggeri sono quelli in grado di gestire solamente le flessioni morfologiche delle parole, più facili da cogliere, mentre quelli aggressivi sono in grado di gestire anche variazioni derivazionali dei termini, che risultano molto più critiche da trattare;

- *stemming* e *lemmatizzazione*: queste due tecniche distinte hanno l'obiettivo di ridurre un lessema alla sua radice (stem), per quanto riguarda il primo, e al lemma associato per quanto riguarda il secondo. La differenza sostanziale tra le due tecniche è che lo stemming include anche le variazioni derivazionali, ed è solitamente più utilizzato in contesti applicativi, come ad esempio nell'IR;

- *corpus* (plurale *corpora*): è una collezione rappresentativa di documenti selezionati ed organizzati, utilizzati per l'analisi linguistica. Vengono utilizzati come modello per una determinata lingua: il loro utilizzo

è fondamentale negli stemmer probabilistici, come verrà mostrato in seguito;

– *lexicon*: o più semplicemente lessico, vocabolario, è l'insieme dei lessemi una certa lingua. Con il termine *lexicon* ci si può riferire sia della totalità dei vocaboli di una lingua, che ad un limitato sottoinsieme, magari specifico di un determinato settore (es. lessico dell'IR). Da non confondere con il dizionario, che invece è l'opera che raccogli tutti lessemi, o più correttamente i lemmi ad essi associati.

3.4 Tipologie di Stemmer

In questa sezione verranno presentate e descritte le varie tipologie di stemmer esistenti, elencandone i limiti, le peculiarità e mettendo in luce gli aspetti caratteristici. Per alcune tipologie verranno riportati anche degli esempi, essendo di particolare importanza e rilievo anche a livello storico.

3.4.1 Stemmer basati su regole

La prima tipologia è quella basata su regole, in inglese *rule-based* stemmer. Questo tipo di algoritmi trasforma i vari lessemi nelle rispettive forme base utilizzando regole predefinite, proprie di ogni idioma. Questo fatto comporta che vengano create soluzioni specifiche per ogni singola lingua trattata e necessita inoltre che si abbia un elevato livello di competenze a riguardo. Generalmente tali regole vengono scritte con l'aiuto di esperti linguisti o almeno di persone madrelingua. Talvolta, questi algoritmi utilizzano ulteriori risorse come dizionari, per la disambiguazione delle parole che sono morfologicamente simili ma hanno significati differenti.

Il vantaggio principale che comporta l'uso di questa tecnica, è che una volta create le regole specifiche di una lingua, non serve apportare ulteriori

modifiche e possono essere applicate a qualsiasi *corpus* di documenti. È anche vero però, che per alcune lingue le cui risorse sono molto limitate, risulta complesso realizzare questo tipo di soluzione.

Per quanto riguarda le prestazioni, i *rule-based* risultano essere molto affidabili, soprattutto nel trattare le regole morfologiche molto complesse, ma come già detto in precedenza hanno molti aspetti negativi, come la dipendenza dalla lingua e la necessità di avere una conoscenza approfondita delle regole linguistiche.

Gli stemmer basati su regole possono essere ulteriormente classificati in tre sottocategorie:

- **Algoritmi forza bruta:** sono la soluzione più semplice adottabile e si basano principalmente su tabelle di lookup per riconoscere la radice di una parola. La tabella viene consultata al fine di trovare la corrispondente variazione morfologica e la radice riconosciuta viene restituita. Un vantaggio rispetto alle tecniche basate sulla rimozione degli affissi (successivamente illustrati) è che sono in grado di cogliere variazioni che ne alterano la struttura morfologica, poiché basta memorizzare nella tabella di lookup l'eccezione relativa alla parola. Una limitazione di questa tecnica è che risulta complesso raccogliere a mano tutte le possibili varianti di una parola, per non parlare dell'ingente quantità di spazio che è necessario per la memorizzazione della tabella.

- **Algoritmi di rimozione degli affissi:** anche questa soluzione risulta essere piuttosto semplice, poiché si basa principalmente sulla rimozione di alcuni caratteri della parola, più in generale prefissi o suffissi. Questi algoritmi fanno uso di elenchi di affissi frequenti per la lingua specifica e di alcune regole relative al contesto per meglio effettuare la rimozione delle parti relative alle variazioni morfologiche. Uno dei limiti di questo approccio è che le radici estratte non sono vere e proprie parole della

lingua trattata, e per certi contesti applicativi questo può essere un limite. Un altro aspetto carente di questa tecnica da considerare, è che spesso viene eseguita un'eccessiva fusione di termini che in realtà non sono tra loro legati nel significato.

- **Stemmer morfologici:** questi sono senza dubbio gli algoritmi più evoluti di questa categoria. Richiedono l'uso di grandi dizionari specifici della lingua trattata, contenenti gruppi di parole organizzati secondo le variazioni sintattiche e semantiche. In generale sono in grado di cogliere ed eliminare variazioni morfologiche sia flessionali che derivazionali e spesso sono anche capaci di gestire un gran numero di eccezioni linguistiche.

Nel corso degli anni sono state presentate diverse soluzioni di stemmer basati su regole, dai primi molto rudimentali basati sul semplice troncamento della parola o la rimozione di un certo numero di caratteri, fino alle soluzioni più sofisticate in grado di ottenere risultati più che discreti. Per quanto riguarda i *rule-based* stemmer, quello che rimane ormai da molti anni il punto riferimento è sicuramente il Porter Stemmer [Porter, 1980], sviluppato da Martin Porter nel 1980, per la lingua inglese. Questo algoritmo sebbene sia piuttosto datato, rimane ancora oggi uno dei più utilizzati e un riferimento importante anche per le nuove tecniche proposte. Di questo stemmer sono state sviluppate anche le varie versioni per le principali lingue europee che portano il nome di Snowball stemmer [Porter, 2001].

3.4.2 Stemmer probabilistici

Questa categoria di stemmer utilizza tecniche di addestramento automatico supervisionato o semi-supervisionato per apprendere autonomamente le regole linguistiche attraverso l'elaborazione di specifici *corpora* o *lexicon*. Questi stemmer infatti non necessitano dell'intervento di esperti di linguistica,

ma raggruppano le parole morfologicamente collegate basandosi unicamente sull'analisi dei dato in input, sfruttando particolari metodi matematici che ne caratterizzano il funzionamento.

Il risultato di uno stemmer probabilistico è quindi strettamente legato ai vocaboli con cui viene addestrato. *Corpus* e *lexicon* più ricchi e ben formulati garantiscono prestazioni migliori, ma l'aspetto più importante, è che un tale algoritmo possa gestire esattamente allo stesso modo lingue differenti, diminuendo nettamente lo sforzo rispetto al creare soluzioni ad-hoc per ogni lingua, come nel caso *rule-based*. Numerosi studi hanno dimostrato che gli stemmer di questo tipo sono in grado di ottenere ottimi risultati, e che sono validi sostituti di quelli basate su regole, in particolare per le lingue con risorse linguistiche molto limitate.

Anche questa categoria può essere suddivisa ulteriormente in altre tre sottocategorie di stemmer, che sono:

- **Stemmer basati sull'analisi di *lexicon*:** utilizzano dei *lexicon* per effettuare l'addestramento, quindi si hanno a disposizione soltanto i lessemi e nessun'altra informazione aggiuntiva riguardo la frequenza o il contesto della parola. Metodi di questo tipo utilizzano tecniche come:
  - calcolo della distanza tra stringhe, attraverso misure di similarità;
  - calcolo della frequenza delle sottostringhe per l'identificazione di suffissi;
  - clustering basato su grafo: questo particolare tecnica è quella utilizzata dall'algoritmo GRAS [Paik et al., 2011] che sarà introdotto nel capitolo successivo.
- **Stemmer basati sull'analisi di *corpus*:** a differenza dei precedenti, in questo caso è possibile sfruttare informazioni aggiuntive relative ai lessemi, come ad esempio il contesto in cui sono presenti o

le co-occorrenze dei termini nei documenti esaminati. Rispetto ai precedenti però, necessitano di *corpus* piuttosto grandi per garantire che le informazioni relative alla co-occorrenza siano affidabili. Le tecniche utilizzate da questa classe di stemmer sono:

  - clustering basato sulla co-occorrenza delle parole;
  - clustering basato sulla somiglianza di distribuzione delle parole;
  - clustering basato sulle caratteristiche lessicali e semantiche.

- **Stemmer basati sull'analisi di *n-grammi*:** l'ultima tipologia di stemmer probabilistici è relativa agli stemmer basati sull'analisi della frequenza di *n-grammi*. A differenza dei metodi precedenti, questi stemmer sono in grado di gestire oltre alle classiche variazioni morfologiche flessionali e derivazionali, anche quelle relative alla composizione di termini e le eccezioni ortografiche.

3.4.3 Stemmer ibridi

Quest'ultima categoria, riguarda gli stemmer che combinano tecniche di vario genere. La combinazione degli approcci permette infatti di aumentare le prestazioni degli algoritmi, poiché tende a bilanciare le parti carenti che hanno i differenti metodi. Fondendo metodi statistici a quelli basati su regole, si possono produrre stemmer più efficienti e che riescono a gestire le varie eccezioni linguistiche tipiche di ogni lingua.

3.5 Metodi di valutazione di uno Stemmer

La valutazione di uno stemmer è stato un argomento da sempre molto discusso. Diversi ricercatori hanno proposto soluzioni e metriche di valutazione per misurare l'efficacia di uno stemmer o il grado di errore da

esso generato. In generale, i metodi di valutazione presenti in letteratura possono essere raggruppati in due principali tipologie: i metodi diretti, ovvero quelli che vanno a misurare le prestazioni direttamente sul risultato prodotto dallo stemmer e i metodi indiretti, che misurano l'efficacia in termini di miglioramento delle prestazioni di un sistema che utilizza quel determinato algoritmo di stemming. Di seguito verranno illustrati entrambi.

3.5.1 Metodi di valutazione diretti

Questi metodi di valutazione classificano uno stemmer in base ai risultati prodotti dall'algoritmo, misurando il tasso di errori generati, di parole correttamente ridotte, e il numero medio di parole associate ad una stessa radice. L'aspetto negativo di questo approccio è che necessita che vengano generate delle collezioni di parole per effettuare i test per ogni specifica lingua trattata, e questo comporta anche un oneroso lavoro manuale.

Vengono ora elencati i principali criteri di valutazione diretta di un algoritmo di stemming.

- **Understemming:** questo tipo di errore si verifica quando la riduzione è inferiore rispetto a quella attesa, ovvero lo stem prodotto non è la vera radice della parola, ma presenta alcuni caratteri in più che non dovrebbero esserci. Al fine di quantificare questo tipo di errore è stato proposto un indice in grado di stimarlo [Paice, 1994], chiamato indice di understemming (*Understemming Index*, UI) definito come $UI = 1 - CI$, dove CI è l'indice di combinazione (*Conflation Index*), ovvero il rapporto tra il numero di coppie di parole che sono correttamente raggruppate rispetto al numero di termini totali.

- **Overstemming:** è esattamente l'opposto rispetto al caso appena presentato. Questo errore si verifica quando la radice di una parola viene erroneamente troncata più di quanto dovrebbe essere fatto, e

porta alla fusione di parole che non sono morfologicamente correlate. Questo inconveniente può essere ovviato introducendo dei vincoli sulla lunghezza minima di uno stem. Analogamente al caso precedente è stata introdotta una metrica per misurare questo tipo di errore, denominata indice di overstemming (*Overstemming Index*, OI) [Paice, 1994], definita come $OI = 1 - DI$, dove DI è l'indice di distinzione (*Distinctness Index*), ovvero il rapporto tra il numero di coppie di parole che non sono state fuse in un unica radice, rispetto al numero di termini totali.

- **Misure di accuratezza:** La precisione di uno stemmer più essere misurata confrontando i risultati prodotti in output dalle diverse tecniche, rispetto alla soluzione ideale attesa. Alcune misure proposte per definire l'efficacia di uno stemmer sono le seguenti [Frakes and Fox, 2003]:

  - **Fattore di compressione dell'indice:** ovvero di quanto viene ridotta la dimensione del *corpus* e quindi dell'indice che verrà generato da esso. Valori elevati di questo parametro denotano un punto di forza dello stemmer;
  - **Numero medio di termini per classi di combinazione:** ovvero la cardinalità media dei raggruppamenti di parole con la stessa radice. Per garantire elevate prestazioni, è auspicabile avere un elevato valore di questa misura;
  - **Rapporto tra numero di parole e stem:** in cui viene semplicemente rapportato il numero di termini trattati e il numero di stem generati. Stemmer efficaci aumentano la differenza tra questi due valori, e incrementano quindi loro rapporto;
  - **Distanza media tra parole e stem associato:** nel quale vengono utilizzate misure di similarità come la distanza di Hamming, ovvero il numero di posizioni nelle quali i simboli corrispondenti sono diversi.

3.5.2 Metodi di valutazione indiretti

I metodi indiretti di valutazione, misurano l'efficacia di uno stemmer attraverso l'analisi delle prestazioni delle applicazioni che utilizzano quello stemmer nella fase di preprocessing, ovvero in termini di aumento delle prestazioni indotto. Il principale vantaggio di questi metodi è che non necessitano di alcun lavoro manuale, ma fanno uso di strumenti automatici per misurare le prestazioni di un sistema, tipici del contesto di utilizzo.

Per quanto riguarda l'Information Retrieval, sono utilizzate le ben note misure delle prestazioni di un sistema di reperimento. Queste misure saranno anche quelle utilizzate nei capitoli successivi:

- ***Precision*** **(Precisione):** è definita come il rapporto tra il numero di documenti recuperati rilevanti rispetto al *topic* (argomento di interesse o query) e il numero totale di documenti recuperati:

$$\text{Precision} = \frac{|Documenti\ rilevanti\ \cap\ Documenti\ recupertati|}{|Documenti\ recupertati|} \tag{3.1}$$

  Alcune varianti interessanti di questa misura sono $P(k)$ (o $P@k$), ovvero la precisione ai primi k documenti recuperati. Nel caso speciale $k =$ *Recall base* si ottiele la misura *R-Prec*, dove la *Recall base* è il numero di documenti rilevanti di un determinato *topic*.

- ***Recall*** **(Richiamo):** è definito come il rapporto tra il numero di documenti recuperati rilevanti e il numero totale di documenti rilevanti (ovvero la *Recall base* del *topic*):

$$\text{Recall} = \frac{|Documenti\ rilevanti\ \cap\ Documenti\ recupertati|}{|Documenti\ rilevanti|} \tag{3.2}$$

- ***F-measure* (o *F-score*):** è definita come la media armonica ponderata fra *Precision* e *Recall* e assume valori nell'intervallo $[0, 1]$.

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

  Questa misura è molto utile per testare l'accuratezza dei risultati perchè permette di combinare le due misure precedenti, che sono quelle di riferimento.

- ***Average Precision* (*AP*):** ovvero il valore medio della precisione calcolata per ogni posizione della *ranked list* prodotta per un determinato *topic*. Questa misura è molto utilizzata nell'IR e anch'essa assume valori compresi tra $[0, 1]$:

$$\text{AP} = \frac{\sum_{k=1}^{n}(P(k) \times \text{rel}(k))}{|Documenti\ rilevanti|} \tag{3.4}$$

  dove $P(k)$ è la precisione alla posizione k e $\text{rel}(k)$ è la funzione indicatrice della rilevanza del documento in posisione k (1 se rilevante, 0 altrimenti).

- ***Mean Average Precision* (*MAP*):** anche quest'ultima misura risulta largamente utilizzata nell'Information Retrieval, e non è altro che la media dei valori delle varie *Average Precision* calcolata per vari *topic* t della collezione in esame:

$$\text{MAP} = \frac{\sum_{t \in T} \text{AP(t)}}{|T|} \tag{3.5}$$

  dove T è l'insieme di tutti i *topic* della collezione. Anche la *MAP* assume valori compresi tra $[0, 1]$.

4

L'algoritmo GRAS

In questo capitolo viene presentato l'algoritmo di partenza, sul quale poi verrà svolto il lavoro principale di questa tesi, ovvero lo studio della riproducibilità di tale algoritmo, che verrà trattato nel capito successivo.

L'algoritmo proposto è GRAS (*GRAph-based Stemmer*), un efficace algoritmo di stemming basato su grafo, indipendente dalla lingua, che utilizza modelli probabilistici per raggruppare le parole di un *lexicon* fornito in input, in classi aventi la stessa radice linguistica, eliminandone così le variazioni morfologiche. Questa soluzione è stata pensata appositamente per l'applicazione nel campo dell'Information Retrieval, in particolare nei sistemi di reperimento ad-hoc.

L'algoritmo è stato proposto da J.H. Paik, M. Mitra e S.K. Parui, nel 2011, e risulta tutt'oggi una delle soluzioni più efficienti per quanto riguarda gli stemmer probabilistici, sia in termini di prestazioni aggiunte nei sistemi di IR che lo utilizzano, che dal punto di vista di efficienza computazionale. I tempi che impiega per effettuare lo stemming, sono considerevolmente inferiori alla maggior parte delle altre soluzioni probabilistiche. La velocità è perciò uno dei principali punti di forza di GRAS.

Le caratteristiche di questo algoritmo sono quindi: aumento indotto delle prestazioni del reperimento, indipendenza dalla lingua trattata e basso sforzo computazionale richiesto, quindi velocità di esecuzione.

Il capitolo è così organizzato: nella prima parte viene presentato l'approccio che sta alla base di questo metodo, fornendo alcuni concetti preliminati ed il lessico per poter comprendere meglio come l'algoritmo opera. Successivamente vengono descritti più approfonditamente i punti chiave dell'algoritmo, riportando lo pseudocodice e descrivendo i passaggi fondamentali. Verranno infine descritti ed analizzati i tre parametri principali dell'algoritmo, che ne determinano fortemente le prestazioni.

Per ulteriori approfondimenti si rimanda all'articolo originale [Paik et al., 2011].

4.1 L'approccio di GRAS

4.1.1 Definizione del problema

Si consideri il seguente problema: dato un insieme di parole distinte, si vuole suddividerle in una collezione di classi, che rappresentano ciascuna un insieme di parole morfologicamente correlate tra loro. La formazione di tali classi può essere fatta attraverso l'analisi dei suffissi linguisticamente validi delle parole trattate, ma ciò richiedere la conoscenza specifica della lingua. Poiché l'obiettivo è quello di ottenere un algoritmo in grado di fare questa operazione indipendentemente dalla lingua trattata, non possono essere utilizzare delle liste predefinite di suffissi linguisticamente validi, tantomeno le complesse regole di flessione e derivazione, anch'esse specifiche della lingua trattata. Quello che può essere fatto invece, è individuare le coppie di suffissi validi, attraverso la loro frequenza nelle parole del *lexicon* a disposizione.

4.1.2 Approccio proposto

L'approccio proposto da questo algoritmo si basa su alcune considerazioni semplici, ma i cui risvolti sono di fondamentale importanza. L'idea di

base è che le parole morfologicamente correlate, in genere, condividono un lungo prefisso comune, e le variazioni linguistiche delle parole generalmente comportano le modifiche dei suffissi che vanno ad accodarsi a questi prefissi comuni. Essendo l'obiettivo primario dello stemming cercare di identificare le parole morfologicamente collegate, per contrarle in quella che dovrebbe essere la rappresentazione comune di queste parole, la considerazione appena fatta potrebbe agevolare molto il processo di raggruppamento di tali parole.

Consideriamo quindi come esempio si partenza due parole $w_1 = ps_1$ e $w_2 = ps_2$, che condividono un prefisso comune p sufficientemente lungo. La coppia $\langle s_1, s_2 \rangle$ formata dai suffissi che rimangono dopo aver rimosso la radice comune p alle due parole iniziali, sono con buona probabilità dei suffissi linguisticamente validi, ma per essere certi di questo, dobbiamo fare un'ulteriore verifica. Per verificare la loro validità da un punti di vista linguistico, è necessario accertarsi che altre coppie di parole che condividono un prefisso comune abbastanza lungo, diverso da p, abbiano la stessa coppia di suffissi. Se tale coppia di suffissi infatti, viene riscontrata per un certo numero di coppie di parole, tale coppia di suffissi si può considerare valida. L'idea innovativa è che i suffissi sono considerati a coppie, e non individualmente. Una volta che le coppie di suffisso candidate sono identificate, si può passare alla ricerca delle parole potenzialmente correlate morfologicamente.

Due parole sono potenzialmente correlate morfologicamente se:

- condividono un prefisso comune non vuoto;
- la coppia dei suffissi ottenuta dalla rimozione di prefisso comune, è una coppia si suffissi candidata, individuata nella fase precedente.

Questo tipo di relazione tra parole, può essere modellata da un grafo, nella quale i nodi sono formati dalle parole in questione, e gli archi collegano coppie di parole che sono potenzialmente correlate morfologicamente. Definiamo nodi adiacenti due nodi che sono tra loro direttamente collegati da un arco.

Siano inoltre definiti nodi *pivot*, i nodi che hanno un gran numero di nodi adiacenti e quindi un gran numero di potenziali parole che potrebbero essere morfologicamente correlate alla parola nel nodo *pivot*.

La considerazione finale di questo approccio, si fonda sul fatto che, i nodi adiacenti ad un nodo *pivot*, se condividono un certo numero di nodi adiacenti con esso, con buona probabilità appartengono alla classe morfologica del nodo *pivot*. Una volta generate le classi di parole morfologicamente correlate, lo stemming viene effettuato associando la radice comune alle parole all'interno di una stessa classe.

4.1.3 Terminologia

Di seguito verranno introdotti alcuni concetti utili a comprendere i passi che l'algoritmo svolge per effettuare lo stemming dei termini. Sia L un *lexicon* contenente le parole relative ad un *corpus*. Definiamo:

- *Coppia di suffissi co-occorrente*: una coppia di suffissi $\langle s_1, s_2 \rangle$ è detta *co-occorrente* se esiste una coppia di parole $\langle w_i, w_j \rangle$ in L, tali che $w_i = rs_1$, $w_j = rs_2$, dove r è il più lungo prefisso comune tra w_i e w_j, con $|r| > 0$. Più in generale, i suffissi s_1 e s_2 sono considerati una coppia *co-occorrente* se entrambi possono essere aggiunti in coda ad una radice r per formare delle parole valide w_i e w_j. La coppia di parole $\langle w_i, w_j \rangle$ induce quindi la coppia di suffissi $\langle s_1, s_2 \rangle$, e il numero di coppie di parole che induce una certa coppia di suffissi $\langle s_1, s_2 \rangle$ è chiamato frequenza della coppia di suffissi. Si noti inoltre che, al più uno dei suffissi che formano la coppia $\langle s_1, s_2 \rangle$ può essere $NULL$, poiché non avrebbe senso considerare la coppia di suffissi $\langle NULL, NULL \rangle$ e la relativa frequenza;

- *Coppia di suffissi α-frequente*: una coppia di suffissi *co-occorrente* $\langle s_1, s_2 \rangle$ è detta *α-frequente* se la sua frequenza è $\geq \alpha$, ovvero se ci sono almeno α parole che utilizzano entrambi i suffissi s_1 e s_2;

– *Parole correlate*: due parole distinte $\langle w_i, w_j \rangle$ si definiscono *correlate* l'una all'altra, se $w_i = rs_1$, $w_j = rs_2$, r è il più lungo prefisso comune tra w_i e w_j, con $|r| > 0$, e la coppia di suffissi $\langle s_1, s_2 \rangle$ è α-*frequente*.

4.1.4 Principi base

I seguenti due principi riguardanti le classi di parole morfologicamente correlate sono la chiave per l'algoritmo proposto:

1. Per ogni classe, esiste una particolare parola chiamata *pivot*, le altre parole della classe sono tutte correlate al *pivot*.

2. Per ogni parola in una specifica classe, eccetto il *pivot*, la maggior parte dei suoi adiacenti, sono anche adiacenti al *pivot* di quella classe.

Il principio (2) in particolare, è fondamentale per il funzionamento dell'algoritmo, poiché permette di fondere nella stessa classe parole che condividono una coppia di suffissi frequenti, solo se esse sono realmente correlate morfologicamente.

Attraverso il solo utilizzo del principio (1), si avrebbe il raggruppamento nella stessa classe di alcune parole che condividono coppie di suffissi ritenuti linguisticamente valide ma che non risultano essere morfologicamente correlate tra loro. Il principio (2) assicura che tale errore non venga commesso e consente a GRAS di risolvere efficacemente alcuni dei più comuni errori commessi da altri stemmer probabilistici.

4.2 Punti chiave dell'algoritmo

Di seguito verrà fatta luce sui tre punti chiave dell'algoritmo che sono l'identificazione delle coppie di suffissi frequenti, la costruzione del grafo pesato e la formazione delle classi di parole morfologicamente correlate.

Verranno descritti i passaggi fondamentali di tali fasi e verrà anche presentato il relativo pseudocodice. Nell'ultima parte verranno fatte alcune considerazioni sui i parametri dell'algoritmo.

4.2.1 Identificazione delle coppie di suffissi frequenti

La prima parte interessante dell'algoritmo GRAS, è quella relativa all'identificazione delle coppie di suffissi e il calcolo della loro frequenza, per decretare quali tra queste coppie siano α-*frequenti* e quindi possano essere utilizzate nel passo successivo per costruire il grafo.

La prima operazione compiuta dall'algoritmo è eseguire una partizione del *lexicon* ricevuto in input, in classi di parole aventi un prefisso comune almeno di lunghezza l. Ovviamente le parole che non avranno un prefisso condiviso con altre, verranno scartate a priori da tutti i passaggi successivi. Questa operazione può essere eseguita con un'unica scansione delle parole in tempo lineare, su un *lexicon* ordinato.

Una volta formata la partizione, viene considerato un gruppo alla volta. Per ogni coppia di parole all'interno di un certo gruppo, viene estratta la coppia di suffissi relativi a tali parole, sottraendo ad esse il più lungo prefisso comune, che chiamiamo r, che risulta sicuramente $\geq l$. Questo passaggio richiede un tempo di esecuzione quadratico nella cardinalità delle partizioni generate, tuttavia il tempo necessario risulta ininfluente poiché tali gruppi non raggiungono mai una cardinalità elevata, anche con valori bassi di l.

Una volta individuate le coppie di suffissi candidate, è necessario calcolare la loro frequenza, scandendo nuovamente le varie coppie di parole per ciascun gruppo. Le coppie di suffissi che avranno una frequenza $\geq \alpha$, saranno le coppie di suffissi α-*frequenti*.

Algorithm 1 Identify the Suffix Pairs

1: Let $S = \{C_1, C_2, \ldots, C_n\}$ be the set of classes of words such that any two words in any particular class have a longest common prefix of length $\geq l$.
2: **for** $i = 1$ to n **do**
3: $C_i = \{w_1, w_2, \ldots, w_m\}$.
4: **for** any two words $w_j, w_k \in C_i, (j < k)$ **do**
5: Output the suffix pair $\langle s_1, s_2 \rangle$ such that $w_j = rs_1$, $w_k = rs_2$, and r is the longest common prefix of w_i e w_j, with $|r| \geq l$.
6: **end for**
7: **end for**
8: Compute the frequency of all suffix pairs.

4.2.2 Costruzione del grafo

Con le informazioni ottenute nella fase precedente, è possibile costruire un grafo $G = (V, E)$, nel quale V è l'insieme delle parole del *lexicon* considerato. Siano $u, v \in V$ due parole, e sia $w(u, v)$ la frequenza della coppia di suffissi indotta dalla coppia di parole associate ai vertici u e v. Se risulta $w(u, v) \geq \alpha$, allora la coppia di suffissi è α-*frequente* ed i vertici u,v sono collegati da un arco pesato uguale alla frequenza $w(u, v)$. Si può quindi definire l'insieme degli archi come $E = \{(u, v)|w(u, v) \geq \alpha\}$. Ovviamente si ha che $w(u, v) = w(v, u)$ e quindi il grafo generato risulta essere non orientato. Il grado di ogni nodo indica il numero di archi incidenti ad esso, indipendentemente dalla loro frequenza. Il termine $Adjacent(v)$ rappresenta l'insieme dei nodi che hanno un arco direttamente collegato a v (si noti che $v \notin Adjacent(v)$).

4.2.3 Formazione delle classi

Una volta costruito il grafo, è possibile passare all'identificazione delle classi di parole morfologicamente correlate. In realtà, le classi vengono costruite sulla base di quelle precedentemente create durante la fase di partizionamento del *lexicon*. Quello che viene fatto è decomporre iterativamente queste classi, o meglio il grafo costruito a partire da esse,

Algorithm 2 Identify Class

while $G \neq \emptyset$ **do**
  Let u be the vertex of G with maximum degree.
  $S = \{u\}$
  for all $v \in Adjacent(u)$ taken in decreasing order of $w(u, v)$ **do**
    if $cohesion(u, v) \geq \delta$ **then**
      $S = S \cup \{v\}$
    else
      Delete the edge (u, v).
    end if
  end for
  Output the class S.
  From G remove the vertices in S and their incident edges.
  Let G' be the new graph.
  $G = G'$
end while

ottenendo così sottoclassi di parole morfologicamente correlate, basandosi sul principio (2) illustrato nella sezione 4.1.4.

Il processo di decomposizione viene effettuato come segue: sia p il nodo *pivot*, ovvero quello con il grado massimo tra i nodi di G. Essendo questo nodo quello con il maggior numero di adiacenti è con buona probabilità un candidato nodo radice per un certo insieme di parole morfologicamente correlate alla parola associata a p. Per ogni nodo v adiacente a p, presi in ordine decrescente di frequenza $w(p, v)$, quindi dando priorità al candidato di maggior rilievo, viene calcolata la misura di coesione tra p e v definita come:

$$coesione(p, v) = \frac{1 + |Adiacenti(p) \cap Adiacenti(v)|}{|Adiacenti(v)|} \tag{4.1}$$

Tale misura, che è compresa tra $[0, 1]$, definisce il grado di correlazione tra le due parole in esame, in particolare se il valore della coesione è maggiore di una certa soglia δ prefissata, la parola associata a v viene assegnata alla

classe morfologica della parola *pivot* p. Se invece il valore risulta inferiore di δ, l'arco che collega le due parole viene eliminato, poiché non vi è correlazione morfologica tra p e v (questo non impedisce che v possa essere eventualmente accorpato ad una classe di qualche altro nodo *pivot*, ad un passo successivo).

Una volta scanditi tutti gli adiacenti del nodo *pivot* in esame, la classe morfologica associata a p è così definita, e tutti i nodi che ne fanno parte, p compreso, vengono eliminati dal grafo, così come tutti gli archi in uscita associati ad essi. Procedendo iterativamente vengono costruite le varie classi, ad ogni passo per un nuovo nodo *pivot* e l'algoritmo termina nel momento in cui non ci sono più nodi *pivot* da esplorare.

4.3 Considerazioni sui parametri

I tre parametri di questo algoritmo l, α e δ possono far variare di molto le prestazioni e il risultato ottenuto. Verranno quindi fatte alcune considerazioni a riguardo.

- l **:** questo parametro è relativo alla lunghezza minima del prefisso che devono avere due parole morfologicamente correlate. La scelta di un valore troppo basso comporta un peggioramento delle prestazioni. Infatti, l'algoritmo tenderebbe a raggruppare parole che condividono un prefisso molto corto e che quindi potrebbero non essere necessariamente correlate a livello morfologico. Allo stesso tempo l'algoritmo peggiorerebbe anche in termini di efficienza, in quando questa scelta comporta la formazione di classi iniziali molto numerose, rendendo elevato il costo per trovare le coppie di suffissi frequenti. Tuttavia, anche un valore troppo alto porta a un calo delle prestazioni, perché, se da un lato renderebbe più accurato lo stemming per le parole molto lunghe, dall'altro taglierebbe a priori un gran numero di vocaboli dalle classi di partenza, peggiorando quindi le prestazioni complessive dello stemming

effettuato. Pertanto, è importante scegliere questo parametro con molta accuratezza, e soprattutto farlo variare a seconda della lingua trattata, poiché la lunghezza media dei vocaboli di un *lexicon* varia da lingua a lingua.

- α **:** la seconda variabile che comporta variazioni alle prestazioni di GRAS è la frequenza di cut-off per le coppie di suffissi. Un valore troppo alto tenderebbe a scartare molte coppie di suffissi che sarebbero invece valide, mentre un basso valore di α farebbe considerare coppie che in realtà non sono morfologicamente valide. È importante considerare però il fatto che, quest'ultimo errore, potrebbe poi venire corretto dal principio (2) di 4.1.4 che impedisce il raggruppamento di parole che non sono correlate, pertanto tra le due possibilità è preferibile scegliere un valore basso di α.

- δ **:** l'ultimo parametro dell'algoritmo è la soglia della coesione tra due nodi adiacenti e può assumere valori compresi tra 0 e 1. Una scelta ponderata di questo parametro porterebbe a considerare valori compresi nell'intervallo $[0.5, 1]$ poiché per valori minori di 0.5, si avrebbe che i due nodi condividono meno della metà degli adiacenti e pertanto risulterebbero difficilmente correlati. Considerare invece valori molto vicini ad 1, potrebbe essere troppo stringente, pertanto un buon compromesso potrebbe essere intorno allo 0.8.

5 Riproducibilità dell'algoritmo GRAS

In questo capitolo viene presentato lo studio della riproducibilità dell'algoritmo GRAS, illustrato nel capitolo precedente. In particolare viene descritta l'implementazione proposta e vengono confrontati i risultati ottenuti elaborando gli stessi dati con i quali gli autori ne hanno testato le prestazioni.

Nella prima parte viene presentata l'implementazione dell'algoritmo che è stata realizzata, descrivendone dettagliatamente i passaggi principali, giustificando le scelte progettuali effettuate e le strutture dati utilizzate.

Successivamente vengono descritti gli strumenti utilizzati nella fase sperimentale, ovvero le collezioni di documenti, gli strumenti software ed il modello di reperimento con il quale sono stati effettuati i test.

In seguito vengono presentati i risultati ottenuti dai test di valutazione, mettendo a confronto i risultati originali presentati nell'articolo di GRAS [Paik et al., 2011], con quelli ottenuti dall'implementazione proposta. Vengono poi discusse le discrepanze e le cause da cui potrebbero derivare.

È di fondamentale importanza descrivere con estrema precisione tutte le fasi e le scelte effettuate, soprattutto nei passaggi in cui gli autori hanno tralasciato alcuni dettagli, poiché l'obiettivo è quello di cercare di ottenere gli stessi risultati riportati dagli autori nella loro soluzione proposta, al fine di stabilire quanto siano riproducibili gli esperimenti che loro hanno eseguito.

5.1 Implementazione dell'algoritmo

L'implementazione proposta è stata realizzata il linguaggio Java, versione 1.8.0, utilizzando come ambiente di sviluppo Eclipse Neon.2.

L'algoritmo riceve in input un file testuale contenente dei vocaboli in ordine lessicografico separati dal carattere "`\n`" e altri tre parametri: l, relativo alla lunghezza minima del prefisso comune che le parole correlate devono avere, α, che corrisponde alla frequenza di cut-off per quanto riguarda le coppie di suffissi frequenti e δ, che indica la soglia delle coesione oltre la quale due parole risultano appartenere alla stesse classe morfologica. L'output prodotto è un file testuale `out.txt` contente righe nel formato:

<parola> \t <stem> \n

Di seguito vengono presentati e descritti i passaggi più rilevanti, fornendone anche le relative parti di codice. Il codice completo è disponibile al link: `https://github.com/giuliobusato/GRAS`

5.1.1 Partizionamento del *lexicon* in classi:

Dopo aver elaborato il *lexicon* fornito in input, ed aver creato l'array `words` di stringhe in ordine lessicografico, ognuna contenente un lessema, viene creata una partizione di parole avente un prefisso comune di lunghezza $\geq l$. Questa partizione in realtà non viene fisicamente creata, ma vengono memorizzati gli indici relativi alla parole che vi appartengono, in particolare vengono memorizzati gli indici della prima e dell'ultima parola della classe. Questa scelta è stata fatta perché ci saranno parole che non faranno parte di nessuna classe, essendo di lunghezza inferiore ad l (di conseguenza non possono avere nessun prefisso condiviso lungo l), oppure parole che non condividono nessun prefisso, e andrebbero a formare una classe contenete soltanto la parola stessa. Tali classi, risulterebbero nella parte successiva

dell'algoritmo ininfluenti poiché la frequenza dei suffissi è calcolata sulle coppie di parole all'interno di una classe, e perciò non verrebbero mai prese in considerazione. Quindi si è scelto di scartare a priori tali parole e le relative classi. Inoltre, non sarebbe corretto considerare soltanto l'indice iniziale poiché potrebbero verificarsi casi in cui ci sono una o più parole scartate tra una classe e l'altra, pertanto è necessario che le classi siano definite da entrambi i loro estremi. Questa scelta permette di ridurre la dimensione del *lexicon* trattato e quindi di velocizzare i passi successivi.

```
1  // return a list of first and last positions of each class
2  public static List<Integer> getPartitions(String[] words, int l)
3  {
4    List<Integer> classes = new ArrayList<Integer>();
5    for (int i=0; i<words.length; i++)
6      if (words[i].length()>=l)
7      {
8        // find the prefix of length l
9        String prefix = words[i].substring(0,l);
10       int count = 0;
11
12       // search the words with the same prefix l
13       for (int j=i+1; j<words.length; j++)
14       {
15         // if the word is shorter than l stop the search
16         if (words[j].length()<l) break;
17
18         // if the prefixes are different stop the search
19         if (!prefix.equals(words[j].substring(0,l))) break;
20         count++;
21       }
22       // a class must contain at least two words
23       if (count>0){
24         classes.add(i);
25         classes.add(i+count);
26       }
27       i = i+count;
28     }
29   return classes;
30 }
```

Codice 5.1. *Metodo che partiziona il lexicon in classi*

Tali indici, vengono memorizzati in una opportuna lista di interi chiamata `classes`, nella quale sono salvate a due a due le posizioni degli estremi di ogni classe. La lista ha dimensione pari al doppio delle classi trovate.

5.1.2 Ricerca delle coppie di suffissi frequenti

Una volta ottenute le varie classi di parole con prefisso comune di lunghezza almeno uguale ad l, si può passare alla seconda parte dell'algoritmo, ovvero la ricerca delle coppie di suffissi frequenti.

Per ogni classe precedentemente trovata, vengono considerate tutte le combinazioni di coppie di parole, e per ognuna di esse viene generata la relativa coppia di suffissi attraverso la funzione `getSuffixPair`, che restituisce una stringa formata dai suffissi ottenuti sottraendo alle due parole il loro più lungo prefisso comune di lunghezza $\geq l$, concatenati da una virgola.

Ogni coppia trovata viene memorizzata in una HashMap la cui chiave è la coppia di suffissi ed il valore è la relativa frequenza. Dal momento che in seguito verranno considerate soltanto le coppie di suffissi α-*frequnti*, è stato scelto di mantenere due distinte HashMap, una chiamata `nonFrequent` e l'altra `frequent`. Ad ogni iterazione una nuova coppia di suffissi viene presa in considerazione, se questa coppia è nuova, ovvero non compare in nessuna delle due HashMap, viene memorizzata in `nonFrequent` con la relativa frequenza uguale ad 1. Se invece la coppia è già presente nella HashMap `nonFrequent` significa che è già stata trovata in precedenza, ma la sua frequenza è $< \alpha$. Dopo aver aggiornato la frequenza, incrementandola di una unità, se tale valore è diventato $= \alpha$ la coppia viene promossa α-*frequnt* e memorizzata in `frequent`, nel caso contrario rimane in `nonFrequent`. Nell'ultimo caso, se la coppia compare già in `frequent`, viene semplicemente aggiornata la relativa frequenza.

Dopo aver analizzato tutte le coppie possibili di suffissi per ogni classe, nella HashMap `frequent` saranno presenti tutte e sole le coppie di suffissi

α-frequnti, con le relative frequenze, che in seguito vengono utilizzate per costruire il grafo pesato. Per quanto riguarda invece le coppie di suffissi non frequenti, non risultano avere nessuna utilità nel seguito dell'algoritmo, pertanto la HashMap relativa viene svuotata per fare spazio in memoria.

```
1  // return the frequent suffix pairs and their frequency
2  public static HashMap<String,Integer> getFrequentSuffixPairs(String
       [] words, List<Integer> classes,int l,int alpha)
3  {
4    HashMap<String,Integer> frequent=new HashMap<String,Integer>();
5    HashMap<String,Integer> notFrequent=new HashMap<String,Integer>();
6
7    // for each class of words
8    for (int i=0; i<classes.size()-1; i=i+2)
9      // find all the pairs in the class
10     for (int j=classes.get(i); j<=classes.get(i+1); j++)
11       for (int k=j+1; k<=classes.get(i+1); k++)
12       {
13         // obtain the suffix pairs from the words
14         String suffixPair = getSuffixPair(words[j],words[k],l);
15         // update the frequency of the suffix pair
16         Integer frequency = frequent.get(suffixPair);
17         if (frequency!=null)
18           frequent.put(suffixPair,frequency+1);
19         else
20         {
21           frequency = notFrequent.get(suffixPair);
22           if (frequency==null) frequency=0;
23           frequency++;
24           if (frequency==alpha)
25           {
26             // the suffix pair become alpha-frequent
27             notFrequent.remove(suffixPair);
28             frequent.put(suffixPair,frequency);
29           }
30           else
31             notFrequent.put(suffixPair,frequency);
32         }
33       }
34   notFrequent.clear();
35   return frequent;
36 }
```

Codice 5.2. *Metodo che trova le coppie di suffissi frequenti*

5.1.3 Ricerca delle classi di parole correlate

Una volta individuate le coppie di suffissi α-*frequnti*, è possibile costruire il grafo su cui identificare i gruppi di parole morfologicamente correlate.

Scandendo una ad una le classi di partenza, relative al partizionamento del *lexicon* in base ai prefissi, viene costruito un grafo G per ognuna di queste classi, attraverso il metodo `buildGraph`. Tale metodo assegna un nodo ad ogni indice di posizione compreso tra gli estremi della classe, la cui parola associata sarà ovviamente la parola che si trova in quella posizione nell'array `words`, e per ogni coppia di nodi (indici) le cui parole associate generano una coppia di suffissi frequente, viene assegnato un arco, di peso pari alla frequenza di tale coppia di suffissi.

Quindi, viene creato un grafo in cui nei nodi ci sono indici anziché parole, per semplificarne la gestione. Creare un grafo di parole infatti sarebbe risultato più oneroso, in quanto in Java le stringhe vengono trattate come oggetti, oltre al fatto che ci sarebbe stata una ridondanza di dati (parole presenti sia nell'array `words` che nei nodi del grafo).

Una volta generato il grafo relativo alla classe in esame, si passa alla fase di costruzione dell'insieme di parole tra loro correlate. Per ogni *pivot* considerato, restituito dal metodo `getNodeWithMaxDegree`, viene definita la relativa classe di termini correlati morfologicamente ed esso, chiamata S, in cui inizialmente compare soltanto il termine *pivot*. Il metodo `getAdjacentList(p)` restituisce la lista di nodi adiacenti a `p` in ordine decrescente di peso degli archi: questo, come già visto, permette di dare priorità ai principali candidati. Per ogni nodo visitato, adiacente al *pivot*, viene calcolata la funzione di coerenza attraverso il metodo `getCohesion`: se tale valore è maggiore o uguale della soglia prefissata δ, il nodo adiacente in esame viene inserito nella classe S di termini morfologicamente correlati al *pivot*, in caso contrario l'arco viene rimosso attraverso il metodo `removeEdje`.

```
1  //return stem of identified class of morphologically related words
2  public static String[] identifyClass(String[] words, List<Integer>
       classes HashMap<String,Integer> freqSuffPairs,int l,double delta)
3  {
4    String[] stems =  new String[words.length];
5    // for each class of words
6    for (int i=0; i<classes.size()-1; i=i+2)
7    {
8      int first = classes.get(i);
9      int last = classes.get(i+1);
10     // build the graph of the class
11     Graph G = buildGraph(first,last,words,freqSuffPairs,l);
12     // if there isn't any node G.getNodeWithMaxDegree returns -1
13     while ((G.getNodeWithMaxDegree())>=0)
14     {
15       // let p be the pivot node with maximum degree
16       int p = G.getNodeWithMaxDegree();
17       int[] adjacent_p = G.getAdjacentList(p);
18       // let S the class of words morphologically related to p
19       List<Integer> S = new ArrayList<Integer>();
20       S.add(p);
21       // visit adjacent in decreasing order of edge weight
22       for(int v=0; v<adjacent_p.length; v++)
23       {
24         int[] adj_p = G.getAdjacentList(p);
25         int[] adj_v = G.getAdjacentList(adjacent_p[v]);
26         // compute the cohesion between p and v
27         if (getCohesion(adj_p,adj_v)>=delta)
28           S.add(adjacent_p[v]);
29         else
30           G.removeEdge(p,adjacent_p[v]);
31       }
32       // compute the stem for the class S
33       String stem = getStem(S,words,first,l);
34       // remove from G all the vertices in S and their edges
35       for (int j:S)
36       {
37         stems[first+j]=stem;
38         G.removeNode(j);
39       }
40     }
41   }
42   return stems;
43 }
```

Codice 5.3. *Metodo che identifica le classi di parole correlate*

Al termine della scansione degli adiacenti, l'insieme S generato sarà quindi la classe di termini morfologicamente correlati, che vengono mappati in un unico stem, calcolato con il metodo `getStem`, che trova il più lungo prefisso comune a tutti le parole dell'insieme S. Tutti i nodi dell'insieme S e gli archi ad essi incidenti, vengono quindi eliminati dal grafo, attraverso il metodo `removeNode`, ed il processo viene iterato fino a che esiste un nodo *pivot* nel grafo. Una volta svuotato il grafo, si passerà alla costruzione di un altro grafo relativo alla successiva classe di parole.

Si noti quindi che viene costruito un grafo G specifico per ogni classe, relativo cioè ad un ristretto gruppo di parole, ma con le informazioni relative alla frequenza dei suffissi che fanno riferimento a tutto il *lexicon*. Creare un unico grafo di grandi dimensioni non sarebbe appropriato poiché parole relative a classi differenti non possono mai essere correlate morfologicamente.

5.2 Strumenti utilizzati

In questa sezione vengono presentati e descritti brevemente gli strumenti utilizzati per effettuare gli esperimenti, in particolare gli strumenti software, il modello di reperimento utilizzato, le misure di valutazioni e le collezioni sulla quale sono stati eseguiti training e test.

5.2.1 Strumenti software

Per effettuare l'indicizzazione e la fase di reperimento è stato utilizzato Terrier[1] v4.1, un efficiente sistema IR open source sviluppato in linguaggio Java dall'Università di Glasgow. Per poter effettuare lo stemming con l'algoritmo realizzato, è stata apportata una modifica al file sorgente di Terrier, poiché la versione originale permette di utilizzare soltanto alcuni stemmer predefiniti.

[1] http://terrier.org/

L'aspetto più importante di questo strumento è sicuramente il file `terrier.properties` nel quale vengono definiti tutti i parametri per indicare al sistema come effettuare l'indicizzazione e il reperimento. In seguito vengono fatti dei riferimenti a questo file, ma la spiegazione di questo esula dagli obiettivi del lavoro, quindi per qualsiasi chiarimento si rimanda alla documentazione ufficiale[2] che descrive dettagliatamente ogni aspetto.

Per quanto riguarda la fase di valutazione è stato utilizzato invece Trec Eval[3] v8.1 che è lo strumento standard utilizzato dalla comunità TREC per la valutazione ad hoc. Questo programma riceve in ingresso le *run* prodotte da Terrier, ovvero i risultati del reperimento, e i pool dei topic associati, producendo in output tutte le misure di valutazione standard dell'IR.

5.2.2 Modello e metodi di valutazione

Il modello di reperimento utilizzato per effettuare gli esperimenti è l'IFB2 [Amati and van Rijsbergen, 2002], un modello probabilistico basato sulla misura della divergenza della casualità.

Le misure di valutazione utilizzate saranno *MAP* (*Mean Average Precision*) per la fase di training, e *MAP*, *R-Prec*, *P@5*, *P@10* e numero di documenti rilevanti recuperati (*Rel.Ret.*) per la fase di test. Per maggiori dettagli su tali misure si rimanda alla relativa sezione 3.5.2 .

5.2.3 Collezioni sperimentali

Gli esperimenti eseguiti dagli autori di GRAS per valutarne le prestazioni, sono stati condotti su sette lingue differenti: Inglese, Francese, Ungherese, Bulgaro, Ceco, Marathi e Bengali. Purtroppo però le collezioni a disposizione per riprodurre gli esperimenti non comprendevano Marathi e Bengali e quindi

[2] http://terrier.org/docs/v3.6/properties.html
[3] http://trec.nist.gov/trec_eval/

sono stati condotti gli esperimenti soltanto per le prime cinque delle lingue sopraelencate. Nel dettaglio, le collezioni che sono state utilizzate sono:

- **Inglese:** per la fase di test sono stati utilizzati TIPSTER disk4-5 relativi alla campagna TREC-7 [Voorhees and Harman, 1998], comprensivi di *Foreign Broadcast Information Service (FBIS)*, *Financial Times (FT) 1991-1994*, *LA Times*, sulla quale sono stati valutati 150 topic (301-450). Sebbene gli autori dell'articolo abbiano riportato soltanto questi riferimenti, la collezione originale comprende anche i documenti relativi a *Federal Register (FR) 1994* e *Congressional Record (CR) 1993*, quest'ultima che però viene esclusa a priori negli esperimenti anche dagli autori della stessa campagna TREC-7. Per la fase di training è stata utilizzata invece la collezione TIPSTER disk1-2 relativa sempre a TREC, contente *Wall Street Journal (WSJ) 1987-1992*, valutata su 200 topic (1-200). Anche in questo caso quanto riportato non coincide perfettamente con la collezione originale, infatti oltre a *WSJ* tale collezione comprende: *Associated Press newswire (AP) 1988-1989*, *Computer Selects articles Ziff-Davis (ZIFF)*, *Federal Register (FR) 1988-1989* e *abstracts of U.S. DOE publications*.

  Facendo riferimento principalmente ai topic riportati, si è deciso di considerare tutti i documenti relativi, e quindi anche anche questi ultimi, sebbene gli autori non li abbiano citati nell'articolo. Considerare tutti i topic su una collezione parziale a cui essi sono associati sarebbe un grave errore metodologico. L'augurio è che gli autori di GRAS abbiano fatto lo stesso e che siano stati poco rigorosi nel riportare le informazioni relative alle collezioni, altrimenti quanto fatto sarebbe piuttosto grave. C'è però da considerare il fatto che eventuali errori nei risultati ottenuti potrebbero dipendere anche da questo.

- **Francese:** per la fase di training è stata utilizzata la collezione relativa alla campagna CLEF 2004, comprensiva di *LEMOND 1994* e *ATS 1994*,

valutate su 50 topic. Per la fase di test invece sono state utilizzate le collezioni relative a CLEF 2005 e CLEF 2006, comprensive di *LEMOND 1994-1995* e *ATS 1994-1995*, valutate su 100 topic.

- **Ungherese:** sia per la fase di training sia per quella di test è stata utilizzata la collezione *MAGYAR 2002*, che è comune a CLEF per gli anni 2005, 2006 e 2007. Per quanto riguarda i topic, sono stati utilizzati i 50 topic relativi a CLEF 2005 per il training e i 100 topic relativi a CLEF 2006 e CLEF 2007 per il test.

- **Bulgaro:** esattamente come per l'Ungherese, sono state utilizzate le stesse collezioni di documenti per training e test che sono comuni per CLEF relativi agli anni 2005, 2006 e 2007, ovvero *SEGA 2002* e *STANDART 2002*. Per la fase di training sono stati utilizzati i 50 topic relativi CLEF 2005, mentre per quella di test i 100 topic relativi a CLEF 2006 e CLEF 2007.

  Per quanto riguarda questa lingua c'è un'ulteriore considerazione da fare: durante la fase di indicizzazione si è riscontrato che il numero di documenti elaborati non era coerente con quanto riportato dagli autori di GRAS. I documenti riportati relativi a questa collezione sono infatti 87.281, rispetto ai 69.195 trovati. Andando a controllare in un articolo di riferimento per le campagne CLEF [Ferro and Silvello, 2017a], si è riscontrato che il numero corretto di documenti della collezione in esame (tra l'altro l'unica esistente per questa lingua) coincide con quello trovato e non con quello riportato nell'articolo di GRAS. Pertanto, si è concluso che gli autori hanno sicuramente commesso un errore, o un fase di scrittura o in fase di sperimentazione, considerando una collezione con 18.086 documenti in più la cui origine non è ben chiara.

- **Ceco:** per questa lingua invece è stata effettuata soltanto la fase di test, poiché le query a disposizione non erano sufficienti per fare anche la fase

di training. La collezione con cui sono stati effettuati i test sono *LIDOVE 2002* e *MLADA 2002* relative a CLEF 2007, valute su i relativi 50 topic.

5.3 Estrazione dei *lexicon*

La prima fase riguarda l'indicizzazione dei documenti relativi alle collezioni appena descritte, al fine di estrarre dai documenti i vocaboli per generare i *lexicon* da dare in input all'algoritmo di stemming.

Per ogni lingua è stata utilizzata la relativa stoplist tra quelle prodotte da Jacques Savoy, che risultano essere le più diffuse e anche le più accurate; tutte le stoplist sono scaricabili al sito di riferimento[4]. Oltre alle stopword, sono stati rimossi anche tutti i numeri ed i vocaboli contenenti una o più cifre. Ovviamente in questa fase non è stato utilizzato nessuno stemmer in quanto l'obiettivo era quello di produrre il *lexicon* di input per lo stemmer.

In questa fase sono state riscontrate alcune differenze sia relative ai documenti presenti nelle collezioni, sia alle parole indicizzate che vanno a definire il *lexicon* con cui è stato "allenato" l'algoritmo. In particolare, come è già stato detto, si è evidenziata un'evidente anomalia per i documenti della collezione relativa al bulgaro. La differenza è rilevante perché sono risultati 18.086 documenti in meno (circa il 20%) rispetto a quanto riportato nell'articolo di GRAS. Ovviamente da questo dipende anche un'importante differenza per quanto riguarda le parole indicizzate, che risultano quasi il 9% in meno. Questo aspetto ha ovviamente influito nei risultati di tutte le fasi successive, ma si può affermare con buona probabilità che questa anomalia dipenda da un errore commesso degli autori di GRAS.

Per quanto riguarda tutte le altre lingue, il numero dei documenti risulta coerente, ma ci sono più o meno evidenti differenze per quanto riguarda il numero di vocaboli indicizzati. Per il ceco la differenza risulta totalmente

[4] `http://members.unine.ch/jacques.savoy/clef/`

trascurabile, mentre per l'ungherese risulta dell'1.3%, quindi può considerarsi poco influente. Anche per l'inglese l'errore può considerarsi non troppo influente, essendo la differenza del numero dei vocaboli intorno al 3.8%. C'è però da sottolineare che tale risultato è stato ottenuto aggiungendo il parametro `TrecDocTags.skip=PROFILE` nel relativo file properties di Terrier, in quanto l'elaborazione di questo tag in fase di indicizzazione portava ad aumentare in maniera consistente l'indice con parole che non avevano alcun significato morfologico, molto probabilmente codici alfanumerici presenti nei vari documenti della collezione.

Per concludere, anche il francese ha portato a risultati non troppo soddisfacenti in questa prima fase sperimentale, poiché la differenza di vocaboli risulta essere intorno al 7%. Tuttavia per ovviare a questo risultato sono state testate differenti configurazioni dei parametri di indicizzazione, con l'aggiunta o la rimozione di alcuni tag, ma il risultato migliore ottenuto è stato questo, nella quale si sono rimossi tutti i tag dei documenti trattati ad eccezione di `TITLE`, `TEXT`, `TI`, `TX`.

C'è però da considerare il fatto che non necessariamente queste differenze portano ad un peggioramento delle prestazioni, perché potrebbero anche trattarsi di parole che non sono poi utilizzate nelle query e che quindi la loro presenza o assenza non comporta variazioni nei risultati finali.

Nella tabella 5.1 sono riportati i risultati ottenuti in questa prima fase relativa all'indicizzazione dei documenti, in particolare il numero di documenti e parole con le relative differenze riscontrate.

5.4 Tuning dei parametri

Per quanto riguarda il primo parametro di GRAS, ovvero l, gli autori dell'articolo hanno scelto di utilizzare come valore la lunghezza media

Dati riscontrati in fase di indicizzazione

Corpus	Inglese	Francese	Ungherese	Bulgaro	Ceco
Documenti	472 525	177 452	49 530	69 281	81 735
Parole	502 280	325 292	534 813	292 077	457 149

Dati riportati nell'articolo di GRAS

Corpus	Inglese	Francese	Ungherese	Bulgaro	Ceco
Documenti	472 525	177 452	49 530	87 281	81 735
Parole	522 381	303 349	528 315	320 673	457 164

Errore assoluto tra i valori

Corpus	Inglese	Francese	Ungherese	Bulgaro	Ceco
Documenti	–	–	–	18 086	–
Parole	20 101	21 943	6 498	28 596	15

Errore relativo percentuale tra i valori

Corpus	Inglese	Francese	Ungherese	Bulgaro	Ceco
Documenti	–	–	–	20.7215	–
Parole	3.8479	7.2335	1.2299	8.9174	0.0032

Tabella 5.1. *Collezioni di documenti utilizzate per i test*

delle parole della lingua trattata. Come spiegazione risulta tuttavia un po' approssimativa, perché non è specificato se si intende la lunghezza media delle parole di un vocabolario di riferimento, o quelle della collezione esaminata, se la media si intende pesata con la frequenza delle parole o meno. Dopo aver eseguito alcune prove per verificare quale soluzione producesse risultati più simili a quelli ottenuti dagli autori, si è deciso di assegnare ad l il valore della lunghezza media delle parole pesata con la loro frequenza all'interno del *lexicon* trattato, che è risultato essere:

	Inglese	Francese	Ungherese	Bulgaro	Ceco
l	7	7	8	7	6

Gli autori non hanno riportato nessun valore di l a riguardo e quindi è risultato impossibile fare un confronto per tale parametro.

Per quanto riguarda invece gli altri due parametri di GRAS, ovvero α relativo alla frequenza di cut-off delle coppie di suffissi e δ, cioè la soglia del valore di coesione oltre la quale due parole risultano essere correlate morfologicamente, risulta più difficile assegnargli un valore preciso.

Al fine di comprendere meglio il loro impatto sulle prestazioni dello stemming prodotto, gli autori hanno eseguito una fase preliminare di training dell'algoritmo, in modo da capire quali fossero gli effetti delle varie combinazioni di questi parametri e per decretare la configurazione migliore da usare in seguito per eseguire i test sperimentali.

5.4.1 Risultati del training

Di seguito vengono riportati e discussi i risultati ottenuti facendo variare il parametro α tra $\{2, 4, 6, 8, 10\}$ e δ tra $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

Per ogni lingua vengono riportate le tabella dei valori di MAP ottenuti effettuando lo stemmig con i parametri indicati. La prima tabella riporta i risultati ottenuti con la soluzione proposta, la seconda quelli relativi alla soluzione riportata dagli autori di GRAS, la terza è relativa all'errore assoluto tra i due valori precedenti e la quarta indica l'errore relativo percentuale. Nelle ultime due tabelline vengono confrontati i valori medi ed i valori ottimali trovati, entrambi con i rispettivi errori.

Inglese: per quanto riguarda questa lingua i risultati del training, che si trovano in tabella 5.2, risultano essere piuttosto accurati rispetto alla soluzione di riferimento, poiché l'errore medio generato è intorno al 2.3%, quindi entro una soglia di errore sperimentale accettabile. Il fatto che non siano stati definiti con chiarezza molti aspetti già evidenziati in precedenza, comporta inevitabilmente che i risultati non

siano perfettamente coincidenti. L'aspetto forse più interessante di questa lingua è che i valori relativi ai migliori risultati delle due soluzioni esaminate risultano essere molto vicini, in particolare presentano un errore solo dello 0.6%. Si noti che nella maggior parte dei casi, le differenze più consistenti tra i valori risultano essere nelle combinazioni di parametri agli estremi della tabella, in particolare nella prima colonna, quindi in relazione ad un valore basso della soglia di coesione. Questi risultati sono stati ottenuti elaborando nella fase di reperimento solamente il tag `title`, poiché l'aggiunta degli altri tag ne peggioravano le prestazioni complessive.

Francese: i risultati del training di questa lingua, riportati in tabella 5.3, risultano essere leggermente peggiori dei precedenti, in quanto l'errore medio trovato è intorno al 3%. Un fattore potrebbe essere legato sicuramente alla discrepanza già riscontrata nel numero di parole del *lexicon* estratto dalle collezioni. Lo stesso riguarda il risultato relativo al valore ottimale trovato, essendoci un errore leggermente al di sotto di quello precedente. La cosa però rilevante in questa tabella è che i valori dei risultati ottimali sono associati allo stesso valore del parametro δ, il che potrebbe significare che le prestazioni dello stemmer per questa lingua sono dipendenti dal valore che tale parametro assume. Da sottolineare anche il fatto che, come per l'inglese, le prestazioni migliori si ottengono per valori molto vicini all'1 del parametro α. Tali risultati sono stati ottenuti elaborando nella fase di reperimento i tag `title` e `description`, poiché l'aggiunta di `narrative` introduceva rumore, peggiorandone le prestazioni.

Ungherese: per quanto riguarda i risultati di questa lingua, riportati in tabella 5.4, le differenze sono risultate essere sostanziali, poiché l'errore relativo percentuale medio sui valori ottenuti risulta essere del 5.5%, così come

per i valori ottimali trovati. Un aspetto interessante è che il valore ottimale è risultato essere perfettamente coincidente per quanto riguarda la coppia di parametri di stemming. Questo potrebbe indurci ancor di più a pensare, come già evidenziato in precedenza, che le prestazioni relative alla coppia di parametri possano essere in qualche modo legate alla struttura morfologica della lingua. Anche in questo caso i tag utilizzati nel reperimento sono `title` e `description`.

Bulgaro: come si può vedere in tabella 5.5, i risultati ottenuti risultano essere fortemente influenzati dai problemi riscontrati per quanto riguarda i documenti della collezione. L'errore relativo medio è intorno al 9%, e perciò risulta essere molto rilevante. Per quanto riguarda il valore ottimo, c'è un discreto miglioramento, ma in generale i valori risultano essere molto distanti, soprattutto per i valori di δ distanti da 1. Ciò che risalta, però, è sempre il dato relativo al miglior risultato ottenuto, in particolare risulta ancora una volta essere relativo alla stessa combinazione di parametri. Anche per questa lingua i tag selezionati sono `title` e `description`.

Risultati ottenuti con l'algoritmo proposto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.2970	0.2968	0.2966	0.2973	0.2978	0.2986
4	0.2964	0.2970	0.2966	0.2973	0.2989	0.2981
6	0.2962	0.2966	0.2969	0.2968	0.2958	0.2973
8	0.2965	0.2969	0.2966	0.2971	**0.2998**	0.2991
10	0.2964	0.2976	0.2969	0.2974	0.2981	0.2990

Risultati ottenuti con l'algoritmo originale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.2730	0.2840	0.2890	0.2930	0.2930	0.2950
4	0.2820	0.2870	0.2920	0.2970	0.2970	**0.2980**
6	0.2850	0.2880	0.2940	0.2960	0.2930	0.2930
8	0.2850	0.2910	0.2920	0.2930	0.2900	0.2930
10	0.2850	0.2910	0.2920	0.2950	0.2920	0.2910

Errore assoluto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.0240	0.0128	0.0076	0.0043	0.0048	0.0036
4	0.0144	0.0100	0.0046	0.0003	0.0019	0.0001
6	0.0112	0.0086	0.0029	0.0008	0.0028	0.0043
8	0.0115	0.0059	0.0046	0.0041	0.0098	0.0061
10	0.0114	0.0066	0.0049	0.0024	0.0061	0.0080

Errore relativo percentuale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	8.7912	4.5070	2.6298	1.4676	1.6382	1.2203
4	5.1064	3.4843	1.5753	0.1010	0.6397	0.0336
6	3.9298	2.9861	0.9864	0.2703	0.9556	1.4676
8	4.0351	2.0275	1.5753	1.3993	3.3793	2.0819
10	4.0000	2.2680	1.6781	0.8136	2.0890	2.7491

Valore medio

Algoritmo proposto	0.2973
Algoritmo originale	0.2906
Errore assoluto	0.0066
Errore relativo (%)	**2.3184**

Valore ottimale

Algoritmo proposto	0.2998
Algoritmo originale	0.2980
Errore assoluto	0.0018
Errore relativo (%)	**0.6040**

Tabella 5.2. *Risultati dei training relativi all'Inglese*

Risultati ottenuti con l'algoritmo proposto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.3306	0.3306	0.3335	0.3333	0.3318	0.3360
4	0.3318	0.3318	0.3337	0.3322	0.3315	0.3356
6	0.3334	0.3334	0.3334	0.3360	0.3355	**0.3387**
8	0.3340	0.3340	0.3340	0.3347	0.3345	0.3379
10	0.3341	0.3340	0.3345	0.3335	0.3357	0.3376

Risultati ottenuti con l'algoritmo originale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.3190	0.3170	0.3130	0.3280	0.3340	0.3450
4	0.3180	0.3130	0.3230	0.3410	0.3430	0.3410
6	0.3090	0.3130	0.3290	0.3400	0.3370	0.3450
8	0.3150	0.3170	0.3250	0.3390	0.3450	**0.3490**
10	0.3140	0.3210	0.3330	0.3370	0.3340	0.3310

Errore assoluto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.0116	0.0136	0.0205	0.0053	0.0022	0.0090
4	0.0138	0.0188	0.0107	0.0088	0.0115	0.0054
6	0.0244	0.0204	0.0044	0.0040	0.0015	0.0063
8	0.0190	0.0170	0.0090	0.0043	0.0105	0.0111
10	0.0201	0.0130	0.0015	0.0035	0.0017	0.0066

Errore relativo percentuale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	3.6364	4.2902	6.5495	1.6159	0.6587	2.6087
4	4.3396	6.0064	3.3127	2.5806	3.3528	1.5836
6	7.8964	6.5176	1.3374	1.1765	0.4451	1.8261
8	6.0317	5.3628	2.7692	1.2684	3.0435	3.1805
10	6.4013	4.0498	0.4505	1.0386	0.5090	1.9940

Valore medio

Algoritmo proposto	0.3340
Algoritmo originale	0.3289
Errore assoluto	0.0103
Errore relativo (%)	**3.1944**

Valore ottimale

Algoritmo proposto	0.3387
Algoritmo originale	0.3490
Errore assoluto	0.0103
Errore relativo (%)	**2.9513**

Tabella 5.3. *Risultati dei training relativi al Francese*

Risultati ottenuti con l'algoritmo proposto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.3486	0.3484	0.3484	0.3471	0.3523	0.3485
4	0.3488	0.3507	0.3545	**0.3619**	0.3540	0.3452
6	0.3536	0.3570	0.3576	0.3557	0.3493	0.3455
8	0.3554	0.3522	0.3526	0.3520	0.3476	0.3323
10	0.3504	0.3515	0.3521	0.3462	0.3457	0.3301

Risultati ottenuti con l'algoritmo originale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.3340	0.3280	0.3310	0.3310	0.3310	0.3300
4	0.3270	0.3320	0.3330	**0.3430**	0.3330	0.3260
6	0.3310	0.3370	0.3370	0.3350	0.3310	0.3330
8	0.3330	0.3380	0.3370	0.3340	0.3360	0.3120
10	0.3370	0.3350	0.3390	0.3340	0.3350	0.2970

Errore assoluto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.0146	0.0204	0.0174	0.0161	0.0213	0.0185
4	0.0218	0.0187	0.0215	0.0189	0.0210	0.0192
6	0.0226	0.0200	0.0206	0.0207	0.0183	0.0125
8	0.0224	0.0142	0.0156	0.0180	0.0116	0.0203
10	0.0134	0.0165	0.0131	0.0122	0.0107	0.0331

Errore relativo percentuale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	4.3713	6.2195	5.2568	4.8640	6.4350	5.6061
4	6.6667	5.6325	6.4565	5.5102	6.3063	5.8896
6	6.8278	5.9347	6.1128	6.1791	5.5287	3.7538
8	6.7267	4.2012	4.6291	5.3892	3.4524	6.5064
10	3.9763	4.9254	3.8643	3.6527	3.1940	11.1448

Valore medio

Algoritmo proposto	0.3498
Algoritmo originale	0.3317
Errore assoluto	0.0182
Errore relativo (%)	**5.5071**

Valore ottimale

Algoritmo proposto	0.3619
Algoritmo originale	0.3430
Errore assoluto	0.0189
Errore relativo (%)	**5.5102**

Tabella 5.4. *Risultati dei training relativi all'Ungherese*

Risultati ottenuti con l'algoritmo proposto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.2890	0.2875	0.2875	0.2874	0.2833	0.2814
4	0.2885	0.2885	0.2879	0.2893	0.2893	0.2877
6	0.2892	0.2893	0.2893	0.2894	0.2892	0.2897
8	0.2895	0.2895	0.2896	0.2896	0.2897	**0.2898**
10	0.2896	0.2896	0.2892	0.2892	0.2885	0.2838

Risultati ottenuti con l'algoritmo originale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.2530	0.2560	0.2590	0.2610	0.2650	0.2470
4	0.2490	0.2540	0.2640	0.2710	0.2710	0.2740
6	0.2570	0.2570	0.2710	0.2770	0.2690	0.2720
8	0.2590	0.2640	0.2640	0.2760	0.2780	**0.2810**
10	0.2500	0.2600	0.2670	0.2720	0.2730	0.2670

Errore assoluto

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	0.0360	0.0315	0.0285	0.0264	0.0183	0.0344
4	0.0395	0.0345	0.0239	0.0183	0.0183	0.0137
6	0.0322	0.0323	0.0183	0.0124	0.0202	0.0177
8	0.0305	0.0255	0.0256	0.0136	0.0117	0.0088
10	0.0396	0.0296	0.0222	0.0172	0.0155	0.0168

Errore relativo percentuale

$\downarrow \alpha \rightarrow \delta$	0.5	0.6	0.7	0.8	0.9	1.0
2	14.2292	12.3047	11.0039	10.1149	6.9057	13.9271
4	15.8635	13.5827	9.0530	6.7528	6.7528	5.0000
6	12.5292	12.5681	6.7528	4.4765	7.5093	6.5074
8	11.7761	9.6591	9.6970	4.9275	4.2086	3.1317
10	15.8400	11.3846	8.3146	6.3235	5.6777	6.2921

Valore medio

Algoritmo proposto	0.2884
Algoritmo originale	0.2646
Errore assoluto	0.0238
Errore relativo (%)	**9.1022**

Valore ottimale

Algoritmo proposto	0.2898
Algoritmo originale	0.2810
Errore assoluto	0.0088
Errore relativo (%)	**3.1317**

Tabella 5.5. *Risultati dei training relativi al Bulgaro*

5.5 Risultati sperimentali

In questa parte finale verranno discussi i risultati ottenuti in fase di test, confrontando come già fatto in precedenza i risultati riportati dagli autori di GRAS, con quelli ottenuti dall'approccio proposto, evidenziandone le differenze ma soprattutto mettendo in luce quelle che potrebbero essere le cause da cui derivano queste incongruenze.

Per quanto riguarda i parametri utilizzati, gli autori di GRAS hanno considerato come migliore la configurazione $\alpha = 4$ e $\delta = 0.8$, considerando sempre lo stesso valore per l. Sebbene quanto riscontrato in fase di training non sia del tutto coerente questo scelta, è stata adottata la stessa configurazione suggerita, anche e soprattutto per il fatto che l'obiettivo non è ottenere risultati migliori ma cercare di riprodurre quanto è stato fatto.

Per ogni lingua saranno riportate due tabelle, una relativa alla valutazione senza stemmer e la seconda relativa all'uso dello stemmer. Per ognuno dei due casi saranno riportati i differenti valori relativi all'articolo originale e quelli relativi all'implementazione proposta da questo lavoro. Saranno poi riportati come sempre i rispettivi valori relativi ed assoluti, per avere un termine di paragone e poter meglio giudicare i risultati.

In questa fase non verrà considerata soltanto la MAP come in fase di training, ma verrà riportata *MAP*, *R-Prec*, *P@5*, *P@10* e *Rel.Ret.*, ovvero il numero di documenti rilevanti recuperati, al fine di avere più gradi di giudizio dei risultati dello stemming.

Di seguito verranno discussi i risultati delle cinque lingue testate:

Inglese: i risultati ottenuti in fase di test per l'inglese, riportati in tabella 5.6, sono da considerarsi soddisfacenti in quanto l'errore tra i due approcci confrontati sembra essere trascurabile, almeno entro un certo limite di tolleranza. Quello che colpisce è che i risultati non siano pienamente concordanti neppure per quanto riguarda i valori della tabella relativa

Risultati ottenuti senza stemmer

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.2304	0.2763	0.5107	0.4387	6.840
Risultati originali	0.2290	0.2730	0.5000	0.4330	6.812
Err. Ass.	0.0014	0.0033	0.0107	0.0057	28
Err. Rel. (%)	0.6114	1.2088	2.1400	1.3164	0.4110

Risultati ottenuti effettuando lo stemming

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.2749	0.3128	0.5492	0.4859	7904
Risultati originali	0.2700	0.3090	0.5430	0.4790	7873
Err. Ass.	0.0049	0.0038	0.0062	0.0069	31
Err. Rel. (%)	1.8148	1.2298	1.1418	1.4405	0.3938

Tabella 5.6. *Risultati dei test relativi all'Inglese*

alla fase senza stemmer. Questo aspetto mette in luce quanto sia difficile ottenere risultati identici anche utilizzando gli stessi dati e gli stessi strumenti. Spesso si ignora quanto un singolo parametro o una piccola modifica che sembra scontata possa poi portare a lavorare su dati differenti rispetto a quelli di partenza. Considerando quindi fisiologico un errore di questo tipo, possiamo ritenerci soddisfatti dei dati ottenuti per la lingua inglese.

Francese: anche i risultati ottenuti dai test relativi al francese possono considerarsi molto buoni, come si può vedere nella tabella 5.7, in quanto rientrano nella soglia di tolleranza appena discussa sopra. Questo risulta ancora più positivo ripensando all'errore iniziare che aveva dato questa collezione riguardo il numero di parole relative al *lexicon* su cui è stato effettuato lo stemming. Come era già stato accennato, molto probabilmente tra i termini risultavano un gran numero di parole irrilevanti dal punto di vista morfologico, come ad esempio codici, stringhe senza un significato lessicale, che in fase di preprocessing sono

Risultati ottenuti senza stemmer

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.3370	0.3551	0.4751	0.4404	3830
Risultati originali	0.3390	0.3570	0.4750	0.4450	3796
Err. Ass.	−0.0020	−0.0019	0.0001	−0.0046	34
Err. Rel. (%)	−0.5900	−0.5322	0.0211	−1.0337	0.8960

Risultati ottenuti effettuando lo stemming

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.3867	0.3886	0.5495	0.4838	4115
Risultati originali	0.3870	0.3980	0.5330	0.4910	4078
Err. Ass.	−0.0003	−0.0094	0.0165	−0.0072	37
Err. Rel. (%)	−0.0775	−2.3618	3.0957	−1.4664	0.9073

Tabella 5.7. *Risultati dei test relativi al Francese*

state rimosse dagli autori dell'articolo, ma che non hanno specificato nello svolgimento degli esperimenti.

Ungherese: i risultati ottenuti relativi a questa lingua, riportati nella tabella 5.8, non possono ritenersi soddisfacenti. Nella prima tabella i risultati non risultano troppo differenti, addirittura per quanto riguarda i documenti rilevanti recuperati compare lo stesso risultato. Questo significa che la collezione trattata non risulta avere anomalie o documenti mancanti. Il problema è relativo alla parte in cui viene utilizzato lo stemmer proposto, che porta ad un evidente peggioramento rispetto ai risultati riportati dagli autori di GRAS. Ciò che però risulta strano è che in fase di training, i risultati ottenuti dallo stemming con l'algoritmo proposto portano a risultati nettamente migliori rispetto a quelli riportati nell'articolo di GRAS, mentre in fase di test i risultati dell'algoritmo proposto portano a risultati nettamente peggiori, rispetto a quelli di GRAS.

Bulgaro: i risultati relativi al bulgaro, presentati in tabella 5.9, risultano essere

Risultati ottenuti senza stemmer

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.2373	0.2516	0.3531	0.3122	1367
Risultati originali	0.2390	0.2520	0.3570	0.3140	1367
Err. Ass.	−0.0017	−0.0004	−0.0039	−0.0018	–
Err. Rel. (%)	−0.7113	−0.1587	−1.0924	−0.5732	–

Risultati ottenuti effettuando lo stemming

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.3319	0.3467	0.4701	0.4104	1846
Risultati originali	0.3510	0.3600	0.4740	0.4220	1924
Err. Ass.	−0.0191	−0.0133	−0.0039	−0.0116	−78
Err. Rel. (%)	−5.4416	−3.6944	−0.8228	−2.7488	−4.0540

Tabella 5.8. *Risultati dei test relativi all'Ungherese*

perfettamente in linea con quelli del training e con le considerazioni fatte riguardo le collezioni, ovvero risultano completamente sbagliati. Il punto da capire è quali dei due risultati, se quelli relativi a GRAS, o quelli proposti qui siano i più veritieri, poiché essendo le collezioni non coincidenti, i risultati ovviamente non possono risultare concordanti. Ogni altra considerazione per quanto riguarda il bulgaro risulterebbe superflua perché l'errore potrebbe trovarsi proprio all'origine dell'esperimento condotto.

Ceco: sebbene per questa lingua non sia stata effettuata nessuna fase di training, i risultati in fase di test sono risultati sorprendentemente gratificanti. Come si può vedere nella tabella 5.10, i valori relativi alla fase senza stemmer sono risultati perfettamente coincidenti. Per quanto riguarda invece la fase relativa all'utilizzo dello stemmer proposto, la differenza consiste in un solo documento rilevante in più, e tutti gli altri valori risultano pressoché uguali, entro un errore dell'1%.

Risultati ottenuti senza stemmer

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.225	0.2507	0.358	0.284	1521
Risultati originali	0.217	0.229	0.294	0.257	1611
Err. Ass.	0.008	0.0217	0.064	0.027	−90
Err. Rel. (%)	3.6866	9.4759	21.7687	10.5058	−5.5865

Risultati ottenuti effettuando lo stemming

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.341	0.358	0.473	0.372	2043
Risultati originali	0.326	0.334	0.424	0.355	2110
Err. Ass.	0.015	0.024	0.049	0.017	−67
Err. Rel. (%)	4.6012	7.1856	11.5566	4.7887	−3.1753

Tabella 5.9. *Risultati dei test relativi al Bulgaro*

Risultati ottenuti senza stemmer

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.2380	0.2610	0.3160	0.2680	551
Risultati originali	0.2380	0.2610	0.3160	0.2680	551
Err. Ass.	–	–	–	–	–
Err. Rel. (%)	–	–	–	–	–

Risultati ottenuti effettuando lo stemming

	MAP	R-Prec	P@5	P@10	Rel Ret
Risultati ottenuti	0.3630	0.3580	0.4460	0.3720	690
Risultati originali	0.3660	0.3600	0.4480	0.3760	689
Err. Ass.	−0.0030	−0.0020	−0.0020	−0.0040	1
Err. Rel. (%)	−0.8187	−0.5556	−0.4464	−1.0638	0.145

Tabella 5.10. *Risultati dei test relativi al Ceco*

5.6 Considerazioni finali

In questo capitolo si sono riportati i risultati del lavoro di riproducibilità fatto in modo corrispondente a quello eseguito dagli autori di GRAS. L'esperimento è stato riprodotto in tutte le fasi, dall'implementazione dell'algoritmo, alla creazione del *lexicon*, alla fase di training, fino alla fase di test conclusiva.

Le principali difficoltà riscontrate sono state relative a passaggi che gli autori non hanno riportato con la giusta accuratezza, o scelte che sono state fatte senza fornire gli adeguati dettagli. In particolare gli autori sono stati poco precisi nello specificare le collezioni utilizzate, e questo aspetto è molto importante, perché il punto di partenza per riprodurre un esperimento sono proprio i dati su cui esso è stato svolto. Senza gli stessi dati iniziali, ottenere gli stessi risultati risulta pressoché impossibile.

Un altro aspetto a cui gli autori non hanno dato troppo peso sono i parametri, in particolare l, ovvero la lunghezza minima del prefisso delle parole correlate. Cercare di ottenere gli stessi risultati a partire da un algoritmo di cui non si conoscono i parametri, introduce un grado di libertà che complica molto le cose, perché costringe a valutarne in parallelo tutte le possibilità.

Sebbene non sia stato facile riprodurre questi esperimenti, i risultati ottenuti sono stati in parte soddisfacenti. Per alcune lingue i risultati sono risultati buoni, se non ottimi, come ad esempio per il ceco. Le collezione che hanno portato a problemi sono state invece il bulgaro e l'ungherese. Per quanto riguarda il primo la natura delle differenze riscontrate potrebbe addirittura essere un errore degli autori, quindi non risulta corretto penalizzare così tanto i risultati relativi a questa lingua, perché gli errori potrebbero non dipendere dalla qualità del lavoro di riproducibilità quanto alla qualità del lavoro originale svolto. Per quanto riguarda l'ungherese invece, non si è ben capito quale fosse l'origine di un errore così sostanzioso,

poiché lo stesso approccio ha prodotto risultati molto buoni per inglese, francese e ceco, ovvero tutte le altre lingue le cui collezioni risultavano essere compatibili con quelle riportate.

6 Conclusioni

In questa tesi è stata studiata la riproducibilità di GRAS, un algoritmo di stemming che utilizza metodi probabilistici per raggruppare le parole di un *lexicon* in classi morfologicamente correlate, a cui viene assegnata una radice linguistica comune. L'obiettivo era riprodurre quanto svolto dagli autori dell'articolo in cui GRAS è stato presentato, sia per quanto riguarda l'implementazione del codice, sia per la parte relativa alla valutazione sperimentale.

Dopo aver introdotto il concetto di Data-Driven Information Retrieval ed aver spiegato l'importanza della riproducibilità nell'IR, è stata fatta una panoramica sullo stemming, mostrando i vantaggi che introduce l'uso di questa tecnica e descrivendone le principali tecniche e i metodi di valutazione delle prestazioni.

Successivamente è stato presentato l'algoritmo GRAS, evidenziando le caratteristiche peculiari dell'approccio che utilizza per effettuare lo stemming ed analizzando le fasi fondamentali del suo procedimento.

Il lavoro è stato svolto in due fasi distinte: nella prima fase è stata proposta un'implementazione dell'algoritmo GRAS, basandosi sulla descrizione fatta in precedenza. Nella fase successiva si è passati alla valutazione sperimentale della soluzione algoritmica proposta. La parte sperimentale è stata a sua volta suddivisa in una fase preliminare di training dell'algoritmo ed una

successiva relativa ai test finali, per attenersi pienamente a quanto fatto dagli autori. Per ogni risultato ottenuto è stato fatto un confronto diretto con i valori di riferimento presenti nell'articolo, riportando le differenze riscontrate e cercando di individuare la natura delle incongruenze.

I risultati ottenuti in fase sperimentale sono accettabili, in particolare per tre delle lingue prese in considerazione i risultati risultano essere coerenti con quelli presentati nel lavoro originale, pertanto si può considerare l'esperimento riuscito, con un certo margine di tolleranza.

Per le altre due lingue, invece, i risultati sono stati al di sotto delle aspettative. C'è però da sottolineare il fatto che le differenze riscontrate nei risultati riguardanti la lingua bulgara, potrebbero derivare da un errore da parte degli autori. I dati che sono stati riportati relativi alla collezione utilizzata per questa lingua, infatti, non coincidono numericamente con quelli ufficiali della collezione, che sono invece stati riscontrati nel lavoro di riproducibilità. I risultati relativi al bulgaro riportati nell'articolo di GRAS potrebbero non essere veritieri, e se così fosse non avrebbe senso confrontarli con quelli ottenuti da questo lavoro.

La realizzazione di questo lavoro ha permesso di comprendere quanto sia complicato garantire che i risultati di un esperimento siano riproducibili a partire dalla descrizione originale riportata nella presentazioni dei risultati. Spesso vengono dati per scontati o addirittura ignorati aspetti che risultano fondamentali per raggiungere un determinato risultato, o non vengono forniti nel modo adeguato le informazioni relative ai dati di partenza ed i parametri con cui tali dati sono stati elaborati.

Uno degli aspetti di fondamentale importanza nel contesto della riproducibilità di un esperimento, è proprio il tuning dei parametri, che risulta determinante nei risultati prodotti. Riprodurre un esperimento utilizzando gli stessi strumenti e la stessa collezione di dati di partenza non sono condizioni sufficienti per ottenere risultati compatibili con quelli di riferimento.

Riprodurre un esperimento infatti non comporta soltanto la ripetizione di quanto è stato fatto, ma consiste soprattutto nel ricreare il contesto nel quale sono stati ottenuti tali risultati, e ciò risulta molto più complesso, come appare evidente anche dai risultati raggiunti con questo lavoro di tesi.

Bibliografia

[Abadi et al., 2016] Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P. A., Carey, M. J., Chaudhuri, S., Dean, J., Doan, A., Franklin, M. J., Gehrke, J., Haas, L. M., Halevy, A. Y., Hellerstein, J. M., Ioannidis, Y. E., Jagadish, H. V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J. F., Ramakrishnan, R., Markl, V., Olston, C., Ooi, B. C., Ré, C., Suciu, D., Stonebraker, M., Walter, T., and Widom, J. (2016). The Beckman report on database research. *Commun. ACM*, 59(2):92–99.

[Agosti et al., 2016] Agosti, M., Alonso, O., de Rijke, M., and Perego, R. (2016). Data-Driven Information Retrieval. *SIGIR Forum*, 50(2):10–14.

[Allan et al., 2012] Allan, J., Croft, W. B., Moffat, A., and Sanderson, M. (2012). Frontiers, challenges, and opportunities for information retrieval: Report from SWIRL 2012 the second strategic workshop on informationretrieval in Lorne. *SIGIR Forum*, 46(1):2–32.

[Alonso, 2016] Alonso, O. (2016). The Data Stack in Information Retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, page 597.

[Amati and van Rijsbergen, 2002] Amati, G. and van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389.

[Arguello et al., 2015] Arguello, J., Crane, M., Diaz, F., Lin, J. J., and Trotman, A. (2015). Report on the SIGIR 2015 workshop on reproducibility, inexplicability, and generalizability of results (RIGOR). *SIGIR Forum*, 49(2):107–116.

[Croft et al., 2009] Croft, W. B., Metzler, D., and Strohman, T. (2009). *Search Engines - Information Retrieval in Practice*, chapter 8, pages 308–322. Pearson Education.

[Ferro, 2017] Ferro, N. (2017). Reproducibility Challenges in Information Retrieval Evaluation. *J. Data and Information Quality*, 8(2):8:1–8:4.

[Ferro et al., 2016] Ferro, N., Fuhr, N., Järvelin, K., Kando, N., Lippold, M., and Zobel, J. (2016). Increasing Reproducibility in IR: Findings from the Dagstuhl Seminar on "Reproducibility of Data-Oriented Experiments in e-Science". *SIGIR Forum*, 50(1):68–82.

[Ferro and Silvello, 2015] Ferro, N. and Silvello, G. (2015). Rank-Biased Precision Reloaded: Reproducibility and Generalization. In *Advances in Information Retrieval - 37t European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29 - April 2, 2015. Proceedings*, pages 768–780.

[Ferro and Silvello, 2017a] Ferro, N. and Silvello, G. (2017a). 3.5k runs, 5k topics, 3m assessments and 70m measures: What trends in 10 years of adhoc-ish clef? *Inf. Process. Manage.*, 53(1):175–202.

[Ferro and Silvello, 2017b] Ferro, N. and Silvello, G. (2017b). The Road Toward Reproducibility in Science The Case of Data Citation (in print).

[Frakes and Baeza-Yates, 1992] Frakes, W. B. and Baeza-Yates, R. A., editors (1992). *Information Retrieval: Data Structures & Algorithms*, chapter 8, pages 137–160. Prentice-Hall.

[Frakes and Fox, 2003] Frakes, W. B. and Fox, C. J. (2003). Strength and similarity of affix removal stemming algorithms. *SIGIR Forum*, 37(1):26–30.

[Freire et al., 2016] Freire, J., Fuhr, N., and Ruber, A. (2016). Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041). *Dagstuhl Reports*, 6(1):108–159.

[Gray et al., 2005] Gray, J., Liu, D. T., Nieto-Santisteban, M. A., Szalay, A. S., DeWitt, D. J., and Heber, G. (2005). Scientific data management in the coming decade. *SIGMOD Record*, 34(4):34–41.

[Hey et al., 2009] Hey, T., Tansley, S., and Tolle, K. M., editors (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*, chapter 3, pages 109–172. Microsoft Research.

[Kraaij and Pohlmann, 1996] Kraaij, W. and Pohlmann, R. (1996). Viewing stemming as recall enhancement. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 40–48.

[Krovetz, 1993] Krovetz, R. (1993). Viewing Morphology as an Inference Process. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993*, pages 191–202.

[Lin et al., 2016] Lin, J. J., Crane, M., Trotman, A., Callan, J., Chattopadhyaya, I., Foley, J., Ingersoll, G., MacDonald, C., and Vigna, S. (2016). Toward reproducible baselines: The open-source IR reproducibility challenge. In

Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings, pages 408–420.

[Moral et al., 2014] Moral, C., de Antonio, A., Imbert, R., and Ramírez, J. (2014). A survey of Stemming Algorithms in Information Retrieval. *Inf. Res.*, 19(1):1–14.

[Paice, 1994] Paice, C. D. (1994). An evaluation method for stemming algorithms. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 42–50.

[Paik et al., 2011] Paik, J. H., Mitra, M., Parui, S. K., and Järvelin, K. (2011). GRAS: An effective and efficient stemming algorithm for Information Retrieval. *ACM Trans. Inf. Syst.*, 29(4):19:1–19:24.

[Porter, 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Porter, 2001] Porter, M. F. (2001). Snowball: A language for stemming algorithms. Published online. Accessed 11.03.2008, 15.00h.

[Rajput and NilayKhare, 2015] Rajput, B. S. and NilayKhare, A. (2015). A survey of Stemming Algorithms for Information Retrieval. *IOSR Journals (IOSR Journal of Computer Engineering)*, 1(17):76–80.

[Savoy, 2006] Savoy, J. (2006). Light stemming approaches for the French, Portuguese, German and Hungarian languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 1031–1035.

[Sharma, 2012] Sharma, D. (2012). Stemming algorithms: A comparative study and their analysis. *International Journal of Applied Information Systems*, 4(3):7–12.

[Silvello and Ferro, 2016] Silvello, G. and Ferro, N. (2016). "Data Citation is Coming". Introduction to the special issue on data citation. *Bulletin of IEEE Technical Committee on Digital Libraries, Special Issue on Data Citation*, 12(1):1–5.

[Singh and Gupta, 2016] Singh, J. and Gupta, V. (2016). Text Stemming: Approaches, applications, and challenges. *ACM Comput. Surv.*, 49(3):45:1–45:46.

[Voorhees and Harman, 1998] Voorhees, E. M. and Harman, D. (1998). Overview of the seventh text retrieval conference (TREC-7). In *Proceedings of The Seventh Text REtrieval Conference, TREC 1998, Gaithersburg, Maryland, USA, November 9-11, 1998*, pages 1–23.

Ringraziamenti

Questo lavoro di tesi, è soltanto l'ultimo passaggio di un lungo cammino cominciato molti anni fa, che mi ha permesso di crescere innanzitutto come persona, oltre ad avermi fornito un bagaglio culturale che porterò con me per tutta la vita.

Questo è stato possibile, prima di tutto, grazie al sostegno continuo e incondizionato de miei genitori, che con i loro sacrifici mi hanno dato la possibilità di intraprendere questo percorso accademico. Oltre ad avermi fornito i mezzi per farlo, mi sono sempre stati vicini, insieme ai miei due fratelli Marco e Giovanni, aiutandomi a superare i momenti di maggiore difficoltà che ho affrontato.

Grazie anche a Sophie, la mia ragazza, che nonostante la distanza, negli ultimi due anni è sempre stata al mio fianco e mi ha aiutato ad avere piena consapevolezza delle mie capacità e fiducia in me stesso.

Infine, un ringraziamento particolare va anche alla Professoressa Maristella Agosti e al Dott. Ing. Gianmaria Silvello, che con molta pazienza ed impegno mi hanno assistito nello svolgimento di questo lavoro di tesi.